EDITOR + Assembler

Macrofire

* Einführung in die Assemblerprogrammierung
* Spriteeditor
* Programmbeispiele in 6502 Maschinensprache

H. C. Wagner

für

COMMODORE 64

ISBN 3-88963-189-4

Tegernseer Str. 18, D-8150 Holzkirchen

1. Auflage 1984

Gedruckt in der Bundesrepublik Deutschland – Printed in West Germany – Imprime' en RFA.

Dieses Buch ist eine Produktion des HOFACKER Verlages, Ing. W. Hofacker GmbH. Die Produktion erfolgte unabhängig von der Firma Commodore und ihren weltweiten Niederlassungen.

Der Autor bedankt sich für die Hilfe bei der Erstellung dieses Buches bei:
R. Heigenmoser
W. Hofacker
Chr. Goethe, Übersetzung

H.C. Wagner

EDITOR
Assembler
Macrofire
für
COMMODORE 64

H.C. Wagner

EDITOR
Assembler
Macrofire
für
COMMODRE 64

* Einführung in die Assemblerprogrammierung
* Spriteeditor
* Programmbeispiele in 6502 Maschinensprache

Vorwort

Das Ihnen hier vorliegende Buch soll Ihnen bei der Programmierung in 6502 Assemblersprache helfen. Sie finden eine komplette Anleitung zu einem der leistungsfähigsten Editor-/Assembler-Pakete fuer den Commodore-64. Eine kleine, leicht verständliche Einführung in die Assemblerprogrammierung soll Ihnen den Umgang mit MACROFIRE erleichtern und mit diesem extrem leistungsfähigen Werkzeug vertraut machen.

Am Schluß des Buches finden Sie eine wertvolle und umfangreiche Sammlung von Quellprogrammen für Ihren Editor-/Assembler. Verwenden Sie diese zum Üben und Ausprobieren oder nutzen Sie die Ideen in Ihren eigenen Programmentwicklungen. Der Autor wünscht Ihnen beim Programmieren in 6502 Maschinensprache viel Erfolg und hofft, daß die erworbenen Kenntnisse sich auch in irgend einer Weise positiv auf Ihren Berufsweg auswirken werden.

Inhaltsverzeichnis

Programmieren in Assembler

Die meisten Personal Computer Anwender beginnen heute zuerst damit, ihre Rechner in BASIC zu programmieren. Der Grund dafür ist, daß die meisten PC's, und so auch der Commodore-64, standardmäßig mit einem BASIC-Interpreter in ROM ausgerüstet sind. BASIC ist darüber hinaus auch noch recht weit verbreitet und so lassen sich sogar Programme von anderen Computern, nach gewissen Abänderungen, auf dem eigenen Computer implementieren.

Im Laufe der Zeit jedoch verspürt der BASIC-Programmierer, daß gewisse Programmieraufgaben, in BASIC gelöst, nicht mehr seinen Anforderungen genügen. Er entdeckt die Maschinensprache und mit fortschreitender Programmiererfahrung wendet er diese immer öfter an.

Der Assembler ist hier das richtige Werkzeug, um Maschinensprachenprogramme ohne große Mühe in kürzester Zeit zu erstellen.

Assembler haben eigene Codeworte, die dazu dienen, dem Prozessor Schritt für Schritt mitzuteilen, was er zu tun hat. Der Assembler bringt einige Vorteile und Nachteile gegenüber der Programmierung in einer höheren Programmiersprache, wie z.B. in BASIC.

1. VORTEILE

Assemblersprache ist sehr kompakt, da sie erlaubt, Programme zu schreiben, die den kürzest möglichen Code erzeugen.

Sehr direkt, da Sie sich mit dem Prozessor in seiner Sprache "unterhalten" können. Schnelle Ein-/Ausgabe usw.

2. NACHTEILE

Größere Fehlermöglichkeiten, da die Maschinensprache für den durchschnittlichen Anwender schwerer zu verstehen ist. MACROFIRE jedoch hat hier eine Menge Eigenschaften, die Ihnen das Programmieren in Maschinensprache zur wahren Freude machen kann.

Ein in Assembler geschriebenes Programm läuft nur auf dem Computer, für den es entwickelt wurde. Ein Assemblerprogramm, welches z. B. für den C-64 geschrieben wurde und Bildschirm- sowie Tastaturadressen oder dessen USER-Portadressen enthält, arbeitet nur auf dem C-64 und nicht ohne weiteres auf einem VC-20 oder CBM. Schon garnicht auf einem TRS-80, der obendrein noch eine andere CPU, nämlich den Z80 als Zentraleinheit enthält. Ein einfaches BASIC-Programm, welches keine spezifischen PEEK, POKE oder SYS-Befehle enthält, könnte u. U. auf einem C-64, genau wie auf einem CBM/VC-20 oder TRS-80 arbeiten.

Für diejenigen, die sich noch in die 6502 Maschinensprache einarbeiten müssen, empfehlen wir die Bücher Nr. 118 und Nr. 124 aus dem HOFACKER VERLAG, sowie die vielen anderen, heute auf dem deutschen Markt erhältlichen Einführungen in die 6502 Maschinensprache.

Grundsätzliche Arbeitsweise eines Assemblers

Der eigentliche Assemblerbefehl, der vom Anwender über die Tastatur in den Editor eingegeben wird, besteht grundsätzlich aus einem Mnemonic Wort, welches aus drei Buchstaben zusammengesetzt ist. Insgesamt versteht die 6502 solche 55 Standard-Mnemonics.

Sie finden diese Standardbefehle in der folgenden Tabelle sowie ein zugehöriges Programmiermodell, welches Ihnen den Aufbau der Register veranschaulichen soll.

Die mnemonischen Befehle wurden deshalb eingeführt, um dem Programmierer das Leben etwas leichter zu machen. Der 6502 Prozessor kann diese mnemonischen Befehle nicht verstehen. Es ist die eigentliche Aufgabe des Assemblers, diese mnemonischen Befehle (auch als Operationscode) in die Maschinensprache (OBJECT-CODE) zu übersetzen.

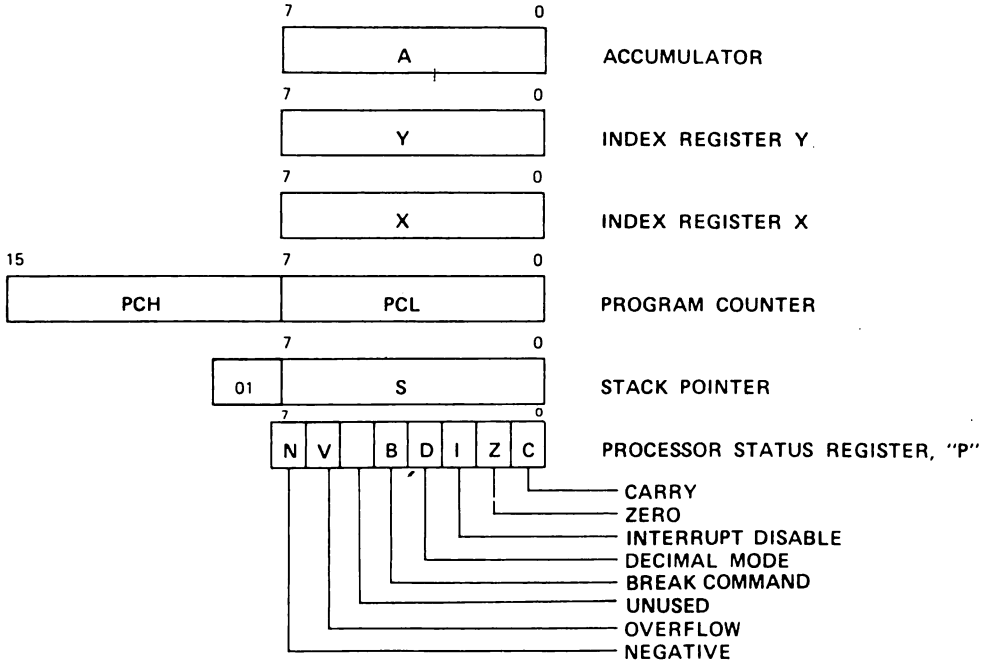
Beispiel: Der Operationscode BPL wird in den Objectcode 10 Hex übersetzt,PHA wird 48 hex usw.

Jeder Operationscode (OPCODE) plus das zugehörige Element mit Daten (Operand) hat einen zugehörigen Objectcode, welcher den Prozessor auf eine bestimmte Speicherzelle dirigiert. Diese Speicherzelle wird auch als Adresse bezeichnet.

R6500 Microprozessorbefehlssatz (alphabetische Reihenfolge)

ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right One Bit (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

Programmiermodell für R6500



Diese Adresse hängt vom jeweiligen Computer ab und kann vom Anwender festgelegt werden. Die Adresse läßt sich leicht vom eigentlichen Objectcode unterscheiden.

Beim C-64 verwenden wir die Adresse C000 hex als Anfangsadresse fuer unsere Maschinenprogramme. Dieser Bereich ist vor dem Zugriff von BASIC geschützt und erlaubt ein problemloses Arbeiten mit Maschinen- und BASIC-Programmen gleichzeitig.

Der Anfang des Maschinenprogrammes wird mit dem "Pseudo OPCODE" ORG festgelegt.

```
ORG  $ C000
```

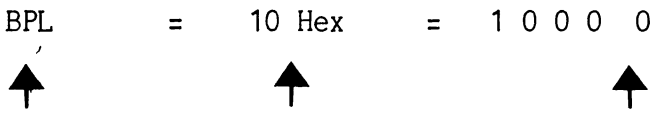
bedeutet z. B. , daß der Programmzähler bei Hexadezimal C000 beginnen soll. ($ ist das Zeichen für Hexadezimal). Der OPCODE in dem darauffolgenden Befehl wird dann in der Speicherzelle C000 Hex abgelegt. Der nächste OPCODE oder OPERAND wird in Zelle 4001 hex. usw. abgelegt. Dies geht dann immer so weiter, bis eine andere Adresse durcheinen ORG-Befehl spezifiziert wird.

Die einzelnen OPCODES sind ausführlich in unserem Buch Nr. 124 beschrieben.

Der 6502 Microprozessor versteht den zugehörigen Objectcode nur in binärer Form. Zur Darstellung auf dem Bildschirm wird der Objectcode zwecks besserer Verständlichkeit in hexadezimale Werte umgewandelt.

Die dezimale Darstellung wird nur selten verwendet. Binär und hexadezimal korrespondieren mehr direkt untereinander und erleichtern so das Arbeiten mit einem Assembler wesentlich.

Jeder Objectcode besteht aus zwei Hexadezimalzahlen oder einem achtstelligen, binären Ausdruck, genannt Byte.



Mnemonic

Objectcode, bestehend aus zwei Hexzahlen

Objectcode, bestehend aus einem achtstelligen, binären Ausdruck.

Wenn Daten eingegeben werden, oder wenn es um die Angabe einer Adresse geht, muß man wissen, welche Schreibweise verwendet werden muß. Der Assembler kann Ausdrucke in dezimal, hexadezimal, binär oder sogar als ASCII-Zahl verstehen. Die folgenden Vorzeichen werden hierfür vom Programmierer verwendet (siehe MACROFIRE-Beschreibung).

%	=	binär
(kein Vorzeichen)	=	dezimal
$	=	hexadezimal

Alle Assembler-Befehle müssen in einer bestimmten Form eingegeben werden. Wenn Sie diese Regeln nicht genau einhalten, werden Sie Fehlermeldungen erhalten.

Die Befehlsformate bestehen in erster Linie aus Assembler-Direktiven (Anweisungen in den Assembler) und Maschinenbefehlen. Die Assembler Direktiven sind Kontrollbefehle an den Assembler und werden auch als Pseudo-Opcodes bezeichnet.

Die am meisten verwendeten PSEUDOOPCODES sind wohl die folgenden:

ORG
EQU
OUT
DFB

(siehe MACORFIRE-Anleitung)

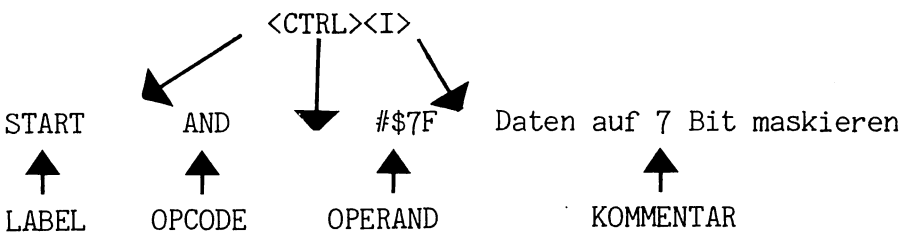
Die Maschinenbefehle enthalten einen der 55 Standardopcodes. Je nachdem, welcher Opcode verwendet wird, kann auch noch ein Operand folgen. Der Operand kann z. B. einen Wert darstellen, an dem der Maschinenbefehl eine Operation vornehmen soll. Weiterhin kann der Assemblerbefehl noch einen Label enthalten. Dieser Label ist eine Marke, die zur einfachen Identifizierung dieses Befehles dienen kann. Am Ende des Befehls kann noch ein Kommentar angebracht werden. Es ist in jedem Falle nützlich, da man später oft vergißt, was man eigentlich durch diesen Befehl erreichen wollte.

Bei Publikationen nutzt es Ihrem Mitmenschen wesentlich, das von Ihnen erstellte Programm leichter zu durchschauen.

MACROFIRE benötigt keine Zeilennummern vor dem Assemblerbefehl, da der Editor voll bildschirmorientiert ist. Sie können sich also mit dem Cursor frei auf dem Bildschirm bewegen und Ihre Eingaben vornehmen.

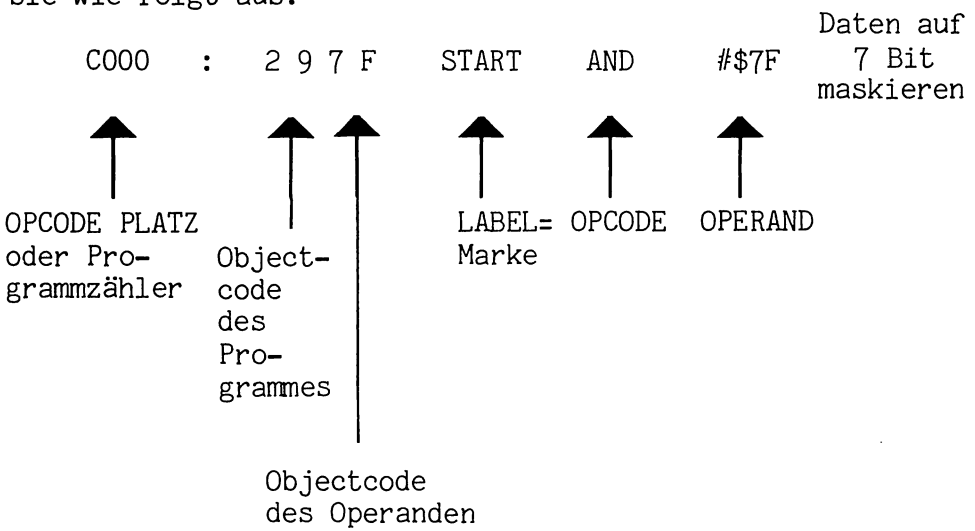
Die Eintragungen der oben besprochenen Elemente können im Editor des MACROFIRE, jeweils getrennt durch ein TAB (<CTRL>-<I>), gemacht werden. Der Kommentar wird einfach hinter den Operanden gesetzt. Sie brauchen keinen Strichpunkt oder ähnliches dazwischenzusetzen. Der Assembler erkennt selbst das Ende des Operanden und weiß, daß alles nach dem Operanden nur Kommentar sein kann. Kommentare werden vom Assembler einfach ignoriert.

Beispiel einer Assembler-Befehlszeile (eingegeben im Editor)



Zwischen den einzelnen Elementen fügen wir ein TAB mit Hilfe des Befehles <CTRL>-<I> ein.

Wenn die Assembler-Befehlszeile später vom Assembler in den eigentlichen Objectcode übersetzt worden ist, sieht sie wie folgt aus:



Da der OPCODE AND in der Speicherzelle mit der Adresse C000 hex liegt, so liegt der Objectcode des Operanden in der darauffolgenden Speicherzelle mit der Adresse C00I Hex. Die nächste Zeile im Programm müßte dann mit der Adresse C002 Hex beginnen und deshalb C002 im Programzählerfeld stehen.

Jeder OPCODE verwendet eine von verschiedenen, festgelegten Vereinbarungen bezüglich der Behandlung von Daten.

Diese festgelegten Vereinbarungen sind auch unter dem Namen "Adressierungsarten" bekannt. Sie sind entsprechend den Datentypen (z.B. Operanden) festgelegt.

Bestimmte OPCODE benötigen überhaupt keinen Operanden, da bei der Befehlsausführung kein bestimmter Wert beteiligt ist, oder der OPCODE eine Operation in sich

selbst definiert und keine weiteren Daten notwendig sind. Bei diesen Opcodes spricht man vom implizierten Adressierungsmodus. Hierzu gehören u.a. der NOP-, CLC- und INX-Befehl. Wenn dem OPCODE ein Operand mit einem konstanten Wert folgt, handelt es sich um unmittelbare Adressierung. Z.B. im Befehl LDA #$60, wobei das #-Zeichen einen konstanten, numerischen Wert darstellt. Das $-Zeichen legt die Zahl als Hexadezimalzahl fest und LDA ist ein Befehl im unmittelbaren Adressierungsmodus.

Wenn jedoch der Operand keine Konstante ist, sondern zu verschiedenen Zeiten unterschiedliche Werte annehmen kann, ist der Opcode im absoluten Adressenmodus. Wenn das obenstehende Beispiel z.B. LDA $ C30A ohne das #-Zeichen gehießen hätte, würde dies heißen, daß jeder Wert, der sich in Adresse $C30A befindet, vom OPCODE entsprechend behandelt wird. LDA wäre dann im absoluten Adressierungsmodus.

Wenn der Wert eines Operanden keine Konstante ist, dann muß es sich um einen Adressenwert aus dem Speicher handeln, und der OPCODE bezieht sich dann auf den jeweiligen Inhalt dieser Adresse und nicht auf den eigentlichen Adressenwert. Die Adressen im Speicher werden in Gruppen, sog. Pages mit je 256 Byte, aufgeteilt. Wenn eine Speicherzelle im Operandenfeld eines Maschinenbefehles spezifiziert wird, besteht diese Adresse aus zwei Bytes. Ein Byte spezifiziert die Page-Zahl, die andere legt das gewünschte Byte in dieser Page fest. Wenn sich die Speicherzelle in der ersten Page des Speichers befindet (Zeropage), wird das Byte, welches die Page spezifiziert, weggelassen (00). Dieser Trick spart Verarbeitungszeit. Wenn sich also ein Operand auf eine Speicherzelle in der Zeropage bezieht, ist der Opcode im Zeropage Adressierungsmodus. Ein Beispiel würde so aussehen:

LDA $0040 (wobei 0040 eine Speicherzelle in der Zeropage ist). Man schreibt den Befehl jedoch gewöhnlich so: LDA $40.

Normalerweise verarbeitet der Prozessor einen Befehl nach dem anderen. Manchmal kann es doch vorkommen, daß der nächste Befehl, abhängig von Daten aus dem vorhergehenden Befehl, ausgeführt werden muß. Es wird eine Entscheidung getroffen, welcher Befehl als nächster ausgeführt werden soll.

In diesen Fällen wird ein Verzweigungsbefehl gegeben, der nur dann ausgeführt wird, wenn eine vorgegebene Bedingung erfüllt worden ist.

Verzweigungsbefehle z.B. BPL und BNE sind OPCODES aus dem relativen Adressierungsmodus. Der dem Verzweigungsbefehl folgende Operand gibt an, welcher Schritt ausgeführt werden soll, wenn die vorgegebene Bedingung erfüllt ist.

Eine andere Art Maschinenbefehl, welche auch den natürlichen Programmablauf unterbricht, ist der JMP-Befehl. Dieses Kommando ist jedoch an keine bestimmte Bedingung gebunden, die erst erfüllt sein muß. Aus diesem Grunde gehört es nicht in den Bereich der relativen Adressierung, sondern ist meist auf die absolute Adressierung festgelegt.

Bei der 6502 Programmierung gibt es noch eine Reihe weiterer Adressierungsarten. Diese sind alle sehr ausführlich in den Büchern Nr. 124 und Nr. 118 beschrieben.

Wir wollen dies hier nicht alles wiederholen, sondern nur einen kurzen Abriss geben und mehr Gewicht auf die Mitteilung von Programmbeispielen legen, die Sie im nächsten Kapitel in großer Auswahl vorfinden werde.

Notizen

Status-Bits und Flags

Innerhalb des 6502 CPM-Systems werden beim Ablauf der Befehle immer wieder Betriebssignale verwendet, die für die interne Steuerung im 6502 Prozessor notwendig sind. Diese Signale nennt man Status-Bits oder auch Flags. Diese Flags sind in einem gesondert dafür eingerichteten Register, dem Statusregister, zusammengefaßt.

Diese Flags beeinträchtigen und beeinflussen den Ablauf der einzelnen Befehle und werden aber auch durch die Ausführung bestimmter Befehle selbst beeinflußt.

Beispiel: Das Carry-Flag wird immer dann gesetzt, wenn zwei Zahlen addiert wurden und das Ergebnis nicht mehr in einem 8 Bit Bytefeld untergebracht werden konnte. Das Zero-Flag wird immer dann gesetzt, wenn das Ergebnis einer Operation gleich Null ist. Das Dezimal-Flag wird immer dann gesetzt, wenn der Programmierer dezimale Werte verwendet. Ein Überlauf-Flag wird dann gesetzt, wenn vorzeichenbehaftete Aktionen oder Subtraktionen durchgeführt werden. Ein Negativ-Flag wird immer dann gesetzt, wenn eine Operation eine negative Zahl liefert.

Wenn ein Flag gesetzt wird, so heißt dies, daß der entsprechende Wert im Register auf logisch 1 gesetzt wird. Wenn die notwendige. zugehörige Bedingung nicht erfüllt wird, wird das Flag zurückgesetzt. (Reset = 0).

Die Verwendung der verschiedenen Adressierungsarten beeinflußt meist eine oder mehrere dieser Flags, welche dann ihrerseits Einfluß auf nachfolgende Befehle innerhalb eines Programmes nehmen können. Die Verzweigungsbefehle richten sich z. B. nach dem Statusbit.

Diese kurze Einführung soll Ihnen die grundsätzlichen Prinzipien der 6502 Assembler Programmierung nahebringen. Wir wollen das bisher Gelernte jetzt noch an einem Beispiel mit dem MACROFIRE demonstrieren.

MACROFIRE - EIN LEISTUNGSFÄHIGES WERKZEUG FÜR DIE PROGRAMMENTWICKLUNG AUF IHREM COMMODORE-64

MACROFIRE ist ein leistungsfähiges Programmentwicklungspaket, bestehend aus drei Programmteilen:

1. Editor
2. Assembler
3. Maschinensprachenmonitor

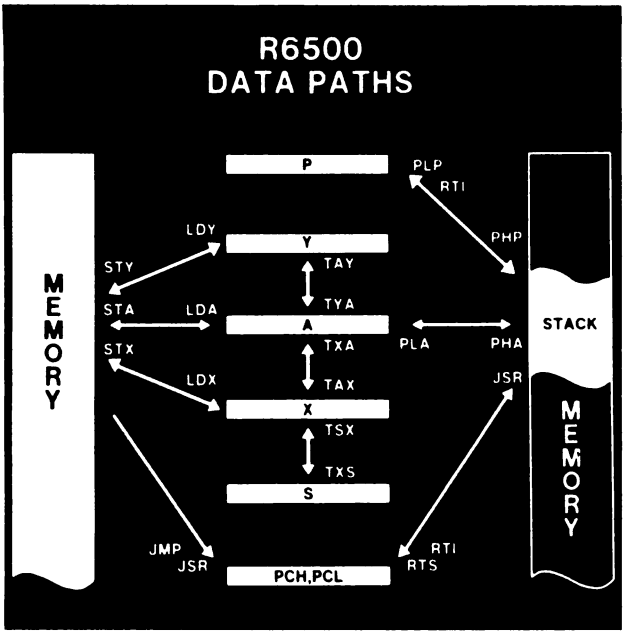
Alle drei Elemente sind integriert, d.h., Sie können jeweils von einem in einen der beiden anderen durch einen einzigen Befehl übergeben. Dies macht MACROFIRE so effizient, wenn es darum geht, leistungsfähige Programme in kürzester Zeit zu entwickeln.

MACROFIRE gibt es auf Cassette oder Diskettte und kann nach dem Laden mit RUN gestartet werden.

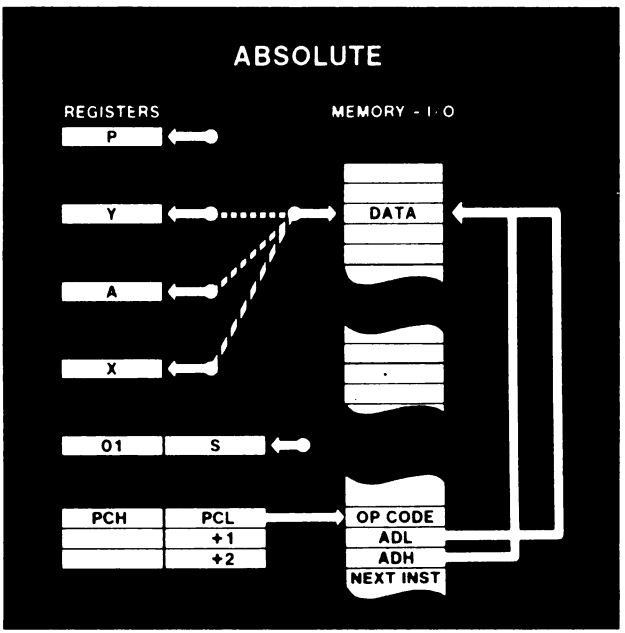
Nach RUN drücken Sie die Space-Taste und Sie befinden sich dann zuerst immer im Editor. Dieser EDITOR dient zur Eingabe Ihres Programmes als Quelltext.

R6500

ADRESSIERUNG

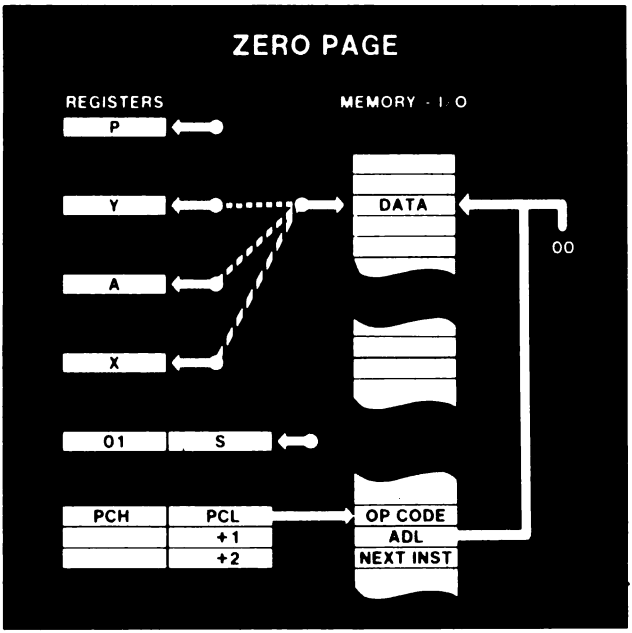


Datenfluß im R6500

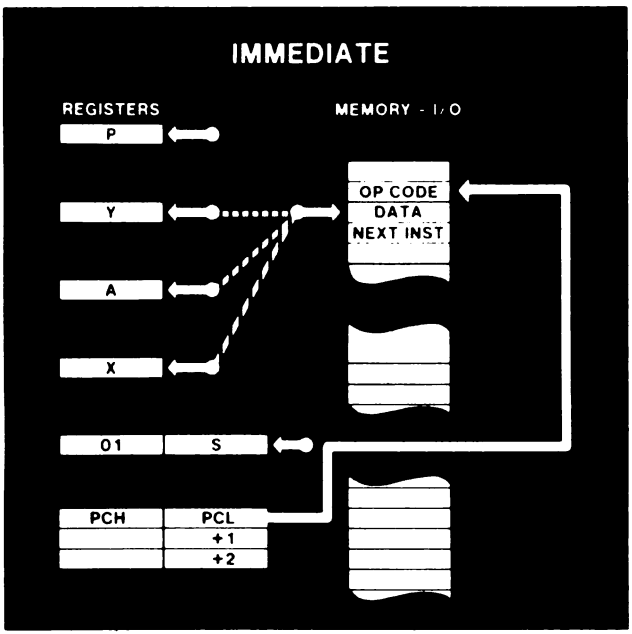


Absolute Adressierung

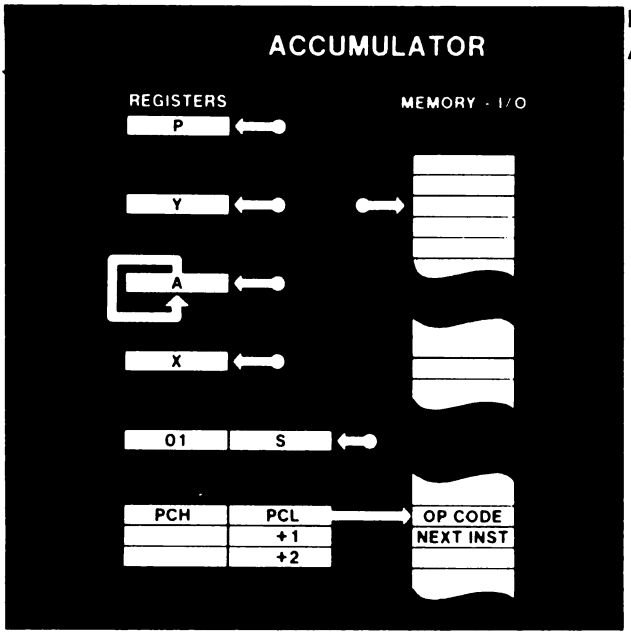
Ein Drei-Byte-Befehl, bestehend aus OPCODE, das zweite Byte enthält das niederwertige Byte der effektiven Adresse ADL. Das dritte Byte enthält das höherwertige Byte der effektiven Adresse (die Adresse, die Daten enthält).
Befehlsimpulse: ADC, DEC, JMP etc.



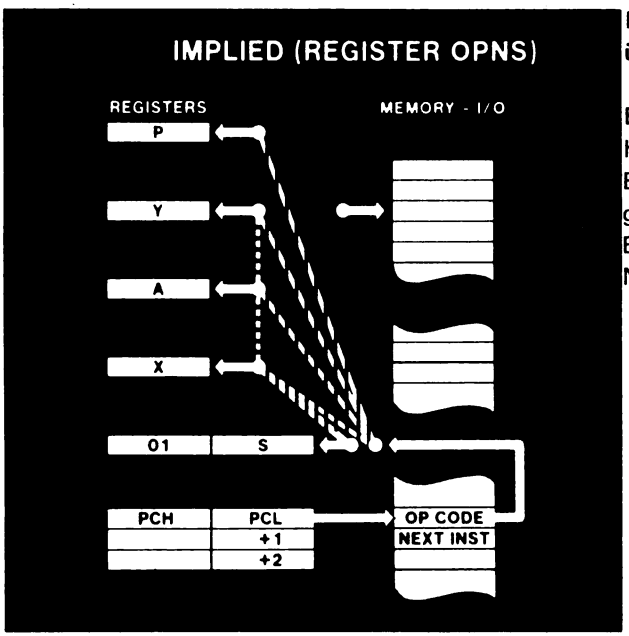
Zero Page Adressierung
(Adressierung über die Page 0 = Speicheranfang)
Zweibytebefehl:
1. Byte OPCODE, zweites Byte enthält die effektive Adresse in der Zero Page (00). Es wird einfach angenommen, daß ADH = 00 ist.
Beispiele: CPX, DEC, LDX etc.



Unmittelbare Adressierung
Zwei-Byte-Befehl:
1. Byte enthält den OPCODE. Im zweiten Byte ist ein Wert, der vom Programmierer vorgegeben wird, abgelegt.
Beispiele: ADC, LDA, LDX etc.
Indirekt, indizierte Adressierung.

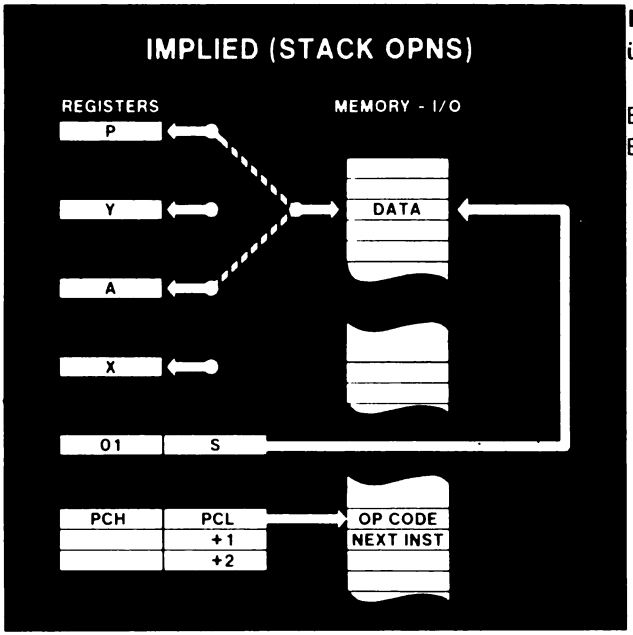


Direkte Manipulation des Akkumulators



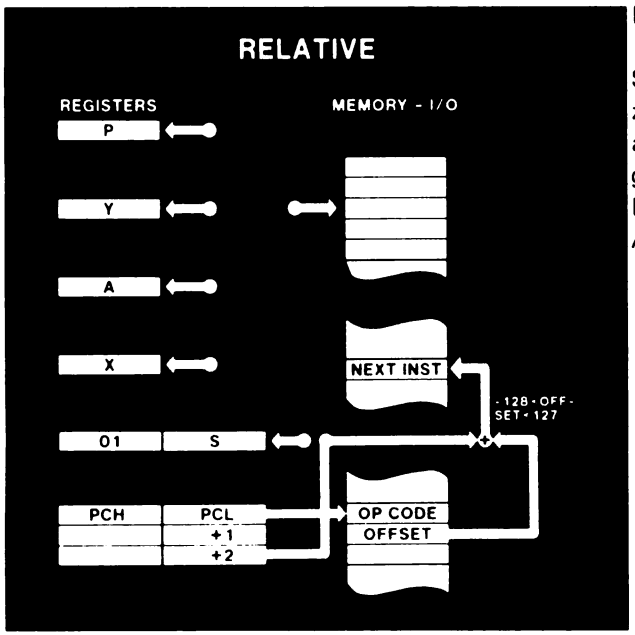
Implizierte Adressierung über Register

Ein Byte-Befehl, innerhalb der CPU werden Bits in den Registern gelöscht oder gesetzt. Beispiele: CLC, INX, NOP, TXA etc.



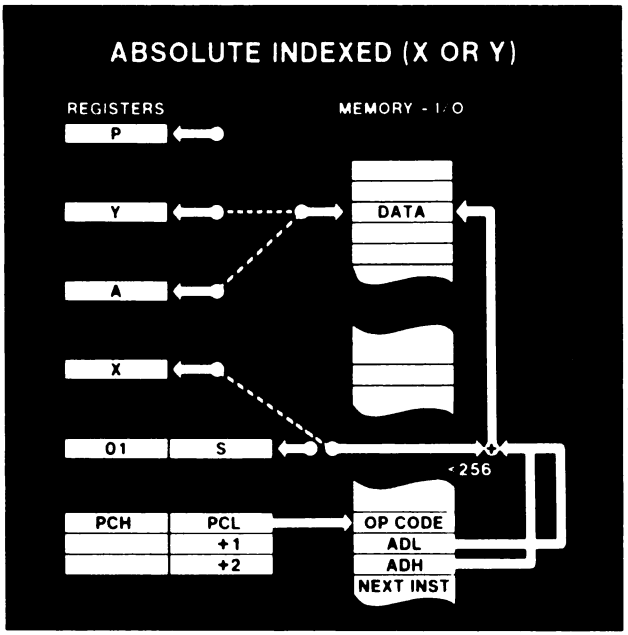
Implizierte Adressierung über Stack

Beispiele: RTS, PLA, BRK



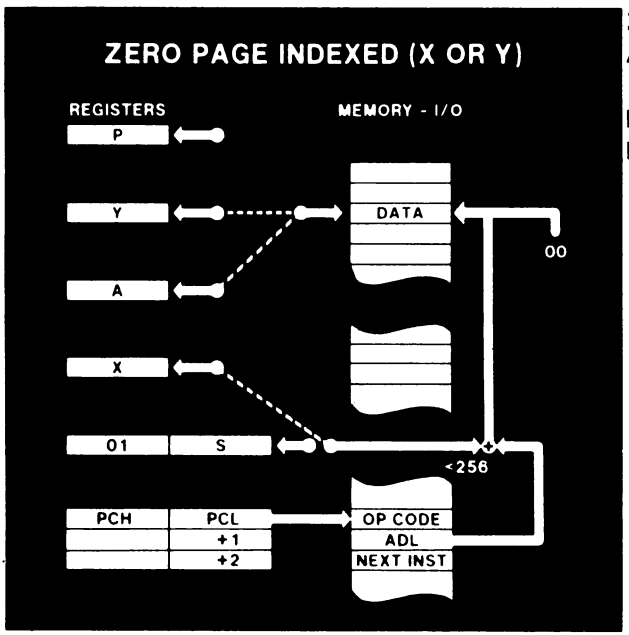
Relative Adressierung

Sie wird meist für Verzweigungsoperationen angewendet. Der Programmzähler wird zur Ermittlung der nächsten Adresse verwendet.



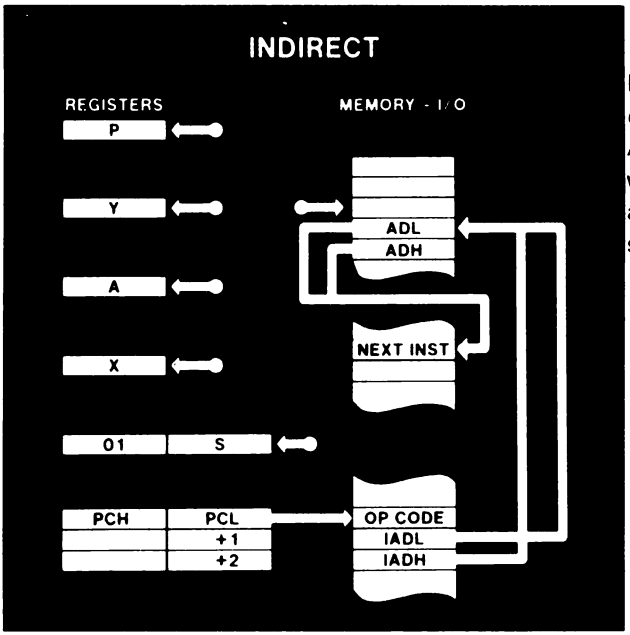
Absolute, indizierte Adressierung

Der Inhalt des X- oder Y-Registers wird zu einer Ausgangsadresse hinzugezählt und so die neue Endadresse ermittelt. Ein Drei-Byte-Befehl. Pagegrenzen beachten.
Beispiel: CPM, ROL, ADC, LDA etc.



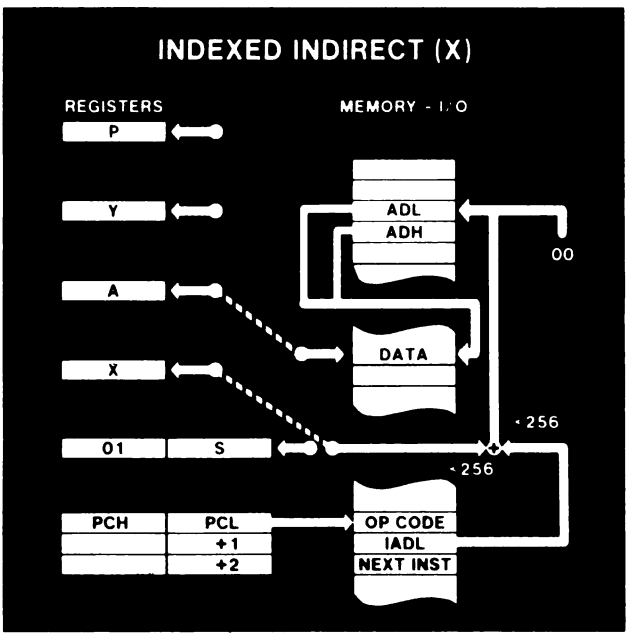
Zero-Page-indizierte Adressierung

Beispiele: AND, CPM, DEC etc.



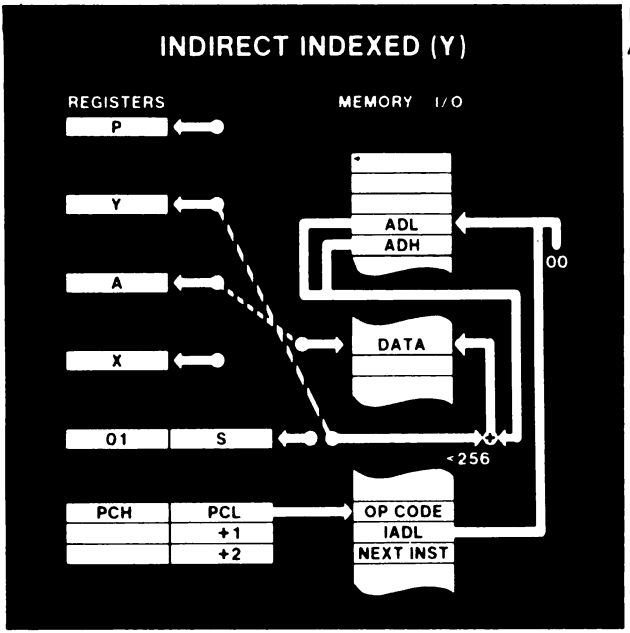
Indirekte Adressierung

Der Befehl besteht aus drei Bytes. Es wird eine Adresse, in der ein gewünschter Befehl steht, aus einer anderen Adresse entnommen.

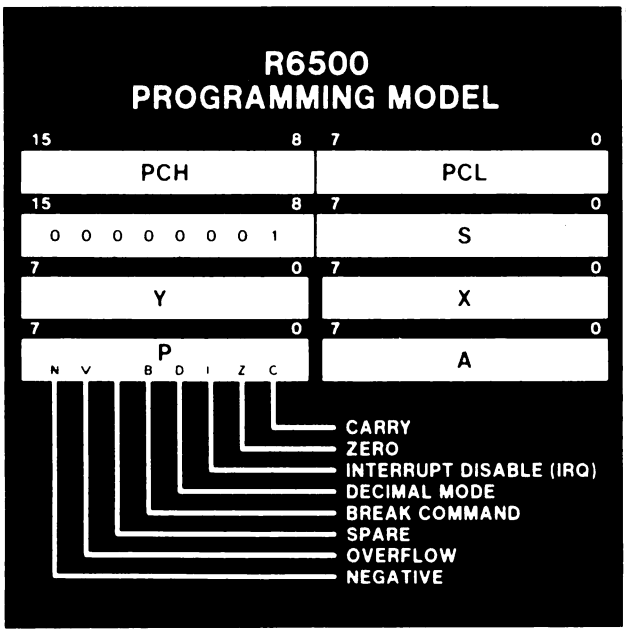


Indizierte, indirekte Adressierung

Der Befehl wird meist für die Abfrage von Ein-/Ausgabegeräten verwendet.
Beispiele: ADC, LDA, STA etc.



Indirekte, indizierte Adressierung



Programmiermodel des R6500

Im Editor, bei dem es sich um ein extrem leistungsfähiges, bildschirmorientiertes Programmsystem handelt, verfügen Sie über eine große Anzahl von Kommandos zur Eingabe und Korrektur Ihres Quelltextes. Automatisches Suchen und Ersetzen, Textblöcke kopieren und löschen, sind nur einige der vielen leistungsfähigen Befehle aus dem Editor. Studieren Sie hierzu die Programmbeschreibung des Editors genau.

Wenn das Quellprogramm im Editor eingegeben worden ist und durch eine Sichtkontrolle Schreibfehler beseitigt wurden, kann der Assembler durch die Eingabe von <CTRL>-<Y> gestartet werden. Nach der Übersetzung wird der Assembler automatisch verlassen und das Programm kehrt wieder in den Editor zurück. Tritt während der Assemblierung ein Fehler auf, so kehrt das Programm sofort in den Editor zurück. Diesmal befindet sich der Cursor jedoch an der Stelle, wo der Fehler aufgetreten ist. Es kann jetzt leicht eine Korrektur durchgeführt und der Assembler neu gestartet werden.

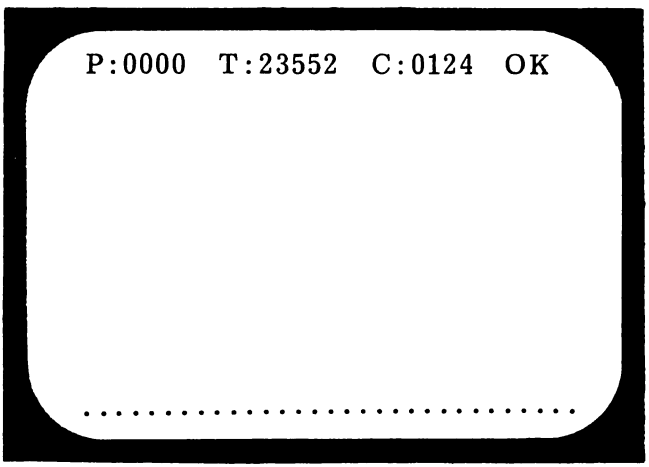
Auf diese Weise ist eine optimale, professionelle Programmentwicklung gewährleistet.

Notizen

Programmierung mit MACROFIRE

1. Erstellung eines Quellprogrammes

Nach RUN und Drücken einer Leertaste sieht der Bildschirm wie folgt aus:



Statuszeile

Kommandozeile

Geben Sie nun folgendes Testprogramm ein und üben Sie damit den Umgang mit MACROFIRE.

```
        OUT LNM,3
        ORG $C000
        LDX #$00 ZÄHLER AUF NULL
START   LDA $0000,X
        JSR $AB47 AUSGABE EINES ZEICH.
        INX       INCREMENTIERE X REG.
        BNE START
        BRK
```

Das obenstehende Programm ist ein einfaches Beispiel. Es druckt einige Zeichen auf dem Bildschirm Ihres C-64 aus. Es dient in erster Linie als Übung zur richtigen Eingabe von Quelltext.

Die <CTRL>-<I> -Funktion für ein TAB, welches uns den Cursor 8 Zeichen nach rechts schiebt, leistet uns hier gute Dienste. Auf diese Weise können wir die OPCODES und Kommentare immer sauber untereinander anreihen.

Wir geben als erstes ein <CTRL>-<I> und rücken den Cursor acht Leerzeichen nach rechts. Dort beginnen also OPCODES und PSEUDO-OPCODES. Als erstes geben wir immer einen OUT Pseudoopcode ein. In unserem Falle OUT LNM, 3. Dieser OUT-Befehl bringt nach der Eingabe von <CTRL>-<Y> ein Assemblerlisting auf den Bildschirm. Siehe hierzu die genauen Anweisungen in der MACROFIRE-Beschreibung.

Wenn wir diesen OUT-Befehl später auf einen Drucker umleiten wollen, so ändern wir diesen einfach mit dem CURSOR in OUT LNM, 4, 0. Dieser Befehl bringt das Assemblerlisting z. B. auf einen angeschlossenen Commodore-Drucker 1525, der auf Device 4 eingestellt ist.

Als zweites geben wir jetzt den ORG-Befehl ein, der die Adresse festlegt, wo unser Maschinenprogramm anfangen soll. ORG $C00. Wichtig ist, daß Sie das Dollarzeichen nicht vergessen und ein Leerzeichen zwischen ORG und dem Dollarzeichen einfügen.

Dann beginnen wir mit unserem eigentlichen Programm und zwar mit dem ersten OPCODE.

Auch hier müssen wir wieder <CTRL>-<I> drücken, um unseren Befehl genau unter den ORG-Befehl zu setzen. Wir geben jetzt LDX #$00 (Leerzeichen) und den Kommentar: ZÄHLER AUF NULL ein.

Die nächste Eingabe ist eine Marke oder auch Label genannt. Sie beginnt gleich am Anfang. Wir geben also nach dem <RETURN> am Ende der vorhergehenden Zeile gleich die Marke START ein. <CTRL>-<I> bringt uns wieder in die Spalte fuer den OPCODE und wir können jetzt LDA $0000,X sehr schön unter den LDX #$00-Befehl schreiben.

Geben Sie auf diese Weise das gesamte Listing ein und speichern Sie es vorerst einmal auf eine vorher formattierte Diskette.

Wie speichert man das Programm auf Diskette?

Gehen Sie mit <CTRL>-<S> mit dem Cursor an den Anfang unseres Quelltextprogrammes.

Drücken Sie dann <CTRL>-<A> und öffnen so die Kommandozeile. Geben Sie in der Kommandozeile W80:TEST1, S,W <CTRL>-<A> <CTRL>-<A> ein. Unser kleines Programm wird dann unter dem Namen TEST1 auf Disk abgelegt.

ASSEMBLIERUNG UNSERES KLEINEN TESTPROGRAMMES

Wenn wir das Programm eingegeben haben und der OUT-Befehl mit OUT LNM,3 auf den Bildschirm gerichtet ist, kann der Assembler durch <CTRL>-<Y> gestartet werden.

Der Bildschirm Ihres C-64 sollte nun folgendes zeigen.

```
                        ORG $C000
C000: A200              LDX #$00 ZÄHLER AUF NULL
C002: B500  START       LDA $0000,X
C004: 2047AB            JSR $AB47 AUSGABE EINES ZEICH.
C007: E8                INX       INCREMENTIERE X REG.
C008: D0F8              BNE START
C00A: 00                BRK

PHYSICAL ENDADDRESS: $C00B

*** NO WARNINGS

START           $C002
```

Das obenstehende Listing wurde mit OUT LNM, 2, 0, "<H86H10> auf einen angeschlossenen Drucker ausgedruckt. Sie können es mit OUT LNM,3 auf Ihrem C-64 Bildschirm ansehen.

Am Ende des Assemblerlistings erhalten wir eine Label-Tabelle.

Das Programm wurde an die Stelle C000 Hex übersetzt und kann dort mit Hilfe des eingebauten Monitors inspiziert werden. Drücken Sie deshalb zweimal die SPACE-Taste und gehen Sie wieder zurück in den Editor. Von hier können wir jetzt auf <CTRL>-<P> den Monitor aufrufen.

Mit MC000 C00F sehen wir uns den Hexcode an und vergleichen diesen mit den Zahlen links im Bild unseres obenstehenden Assembler-Listings. Er sollte identisch sein.

Angenommen wir wollen den Hexcode (Objectcode) auf Diskette speichern, so können wir dies mit

S"TEST2",08,C000,C00F <RETURN>

vom Monitor aus tun. TEST2 ist dann ein kleines Maschinenprogramm, welches Sie später vom Monitor aus mit L"TEXT2", 08 laden können. Von BASIC aus läßt es sich später mit LOAD "TEST2",8, 1 in den Bereich ab C000 Hex laden. Sie könnten es dann von BASIC aus mit SYS49152 starten. Wichtig ist jedoch, daß Sie den BREAK-Befehl am Ende in ein RTS=60 Hex abändern. Dies gilt generell für alle Maschinensprachenprogramme, die Sie von BASIC an aufrufen wollen. Wenn Sie das Testprogramm TEST2 vom Monitor mit G C000 starten wollen, muß ein BRK-Befehl am Ende stehen, der die Kontrolle des Systems nach Ablauf des kleinen Testprogrammes wieder an den Monitor zurückgibt.

Ich empfehle Ihnen jetzt die Eingabe von Quelltextprogrammen etwas zu üben. Versuchen Sie Texte zu ändern, einzufügen und zu löschen.

Ziehen Sie hierzu die Informationen aus der MACROFIRE-Beschreibung zu Rate.

Studieren Sie die jeweils zugehörigen Punkte in der MACROFIRE-Beschreibung.

Wenn Sie einigermaßen sicher sind, gehen Sie dazu über, die Programmbeispiele am Ende des Buches einzugeben und zu testen.

Diese Programmelemente bilden eine wertvolle Sammlung von Programmteilen, die Sie jederzeit in Ihre eigenen Programme einbauen können.

Notizen

Wie verwendet man die HOFACKER-Druckertreiber mit MACROFIRE

Unter der Bestellnummer 4990 bietet der HOFACKER-VERLAG einen Software-Treiber für EPSON- und STAR-Drucker an. Dieser Treiber erlaubt den Betrieb dieser Drucker am C-64, ohne daß eine besondere Hardware erforderlich ist. Alles, was Sie brauchen, ist ein Kabel (11 Leitungen), einen USER-PORT-Stecker, einen 36-Pin Centronics Type-Stecker und natürlich den Software-Treiber. Dieser ist je nach Wunsch auf Disk oder Cassette verfügbar.

Fertigen Sie das Kabel entsprechend der Anleitung an und laden Sie den Treiber nach Anleitung. Starten Sie den Treiber mit RUN und dann mit SYS(52736). Laden Sie dann erst den MACROFIRE und starten Sie MACROFIRE mit RUN.

Die Assemblierung auf den Drucker erfolgt mit

OUT LNM,2,"<CTRL>-<A> H00000 <CTRL>-<A> <CTRL>-<A>

Die Eingabe des OUT-Befehls geschieht, wie gewohnt, am Anfang des Quelltextes. Wenn Sie das Anführungszeichen eingegeben haben, drücken Sie <CTRL>-<A> und erüffnen die Kommandozeile. Dann geben Sie in der Kommandozeile H00000 ein und schließen die Kommandozeile mit ZX <CTRL>-<A> wie gewöhnlich. Jetzt springt der Cursor wieder zurück in den Quelltext. Geben Sie jetzt das Anführungszeichen ein und drücken Sie RETURN.

Zwischen den Anführungszeichen sehen Sie jetzt zwei dieser Zeichen.

Geben Sie <CTRL>-<S> an den Anfang und starten Sie den Assembler. Der Assemblerausdruck erscheint dann nach einer Weile auf Ihrem EPSON- oder STAR-Drucker.

Speicherbelegung MACROFIRE

ZERO P. $02...$7F + was O.S. benutzt

$0800 - $2FFF	Editor + Ass. + Monitor
$3000 - $4FFF	Tabellen für Ass.
$5000 - $53FF	Kopierregister für Editor
$5400 - $AFFF	Textbuffer für Editor
$B000 - $CFFF	Disk I/O im Editor
	Speicherbereich für Anwenderprogramme

Wenn Sie ganz sicher gehen wollen, daß Ihr Anwenderprogramm in keiner Weise ge- oder zerstört wird, so assemblieren Sie in den Bereich ab $B100 - CFFF. Dieser Bereich wird auch nicht von der Disketten I/O verwendet.

Notizen

MACROFIRE – Ein Editor/ Assembler für C-64

WIE VERWENDE ICH DIESES BUCH?

Anweisungen können langweilig, verwirrend und wenig informativ sein, wenn man sich in technische Ausschweifungen verliert (oder?). In diesem Buch wollen wir Erklärungen und Beispiele bringen, die leicht verständlich sind und genau auf einen Punkt zielen. Wir wollen die Komplikationen von programmiertechnischen Ausdrücken vermeiden, aber wenn so etwas doch nötig sein sollte, versuchen wir es so einfach wie möglich und Schritt für Schritt zu erklären.

Wenn Sie ein "Computerneuling" sind, wollen wir darauf hinweisen, daß Sie zuerst die Manuals, die dem C-64 beiliegen, lesen sollten bevor Sie MACROFIRE benützen. Sie werden dann die zahlreichen Befehle von MACROFIRE leichter verstehen, wenn wir diese besprechen.

DER EDITOR

NOTATION

Eine Sache, die bei einem Bedienungshandbuch für einen Computer wohl am meisten frustriert und verwirrt (das

gilt sowohl für den Leser als auch für den Autor) ist die Notation, die verwendet wird, um z. B. anzugeben, was eingegeben (oder eingetippt) werden soll, was der Computer ausdruckt bzw. auf dem Bildschirm darstellt. Unten finden Sie nun die besonderen Notationen mit ihren Erklärungen, die wir in dieser Beschreibung verwendet haben.

<A>

Ein Zeichen oder Wort in Klammeren bedeutet eine Taste Ihrer C-64-Tastatur, in diesem Fall sollte die Taste "A" gedrückt werden.

<CTRL>

Dies bedeutet die "Controll"-Taste auf der linken Seite der Tastatur. In den meisten Fällen wird die <CTRL>-Taste in Verbindung mit einer anderen Taste gedrückt, um MACROFIRE zu veranlassen, eine besondere Funktion auszuführen. Das Zeichen "-" wird dabei benützt, um anzuzeigen, daß zwei Tasten gleichzeitig gedrückt werden sollen. Z.B.

<CTRL>-<A>

bedeutet, daß Sie die <CTRL>-Taste drücken sollen, und während Sie diese gedrückt halten, noch die <A>-Taste drücken sollen. Beachten Sie aber, daß die umgekehrte Reihenfolge nicht das gleiche Ergebnis liefert.

<RETURN>

Diese Taste befindet sich auf der rechten Seite der Haupttastatur. Diese Taste veranlaßt den Cursor an den Anfang der nächsten Zeile zu springen. Sie wird meist benützt, um eine Zeile zu beenden, aber einige Kommandos erfordern auch ein <RETURN> am Ende. Beim Drücken der Taste wird ein "control-M" im Text eingefügt. Normalerweise sind alle Kontroll-Zeichen außer "M" und "I" auf dem Bildschirm sichtbar. Diese beiden Zeichen sind normalerweise nicht sichtbar, weil sie Zeilen beenden oder Leerzeichen anzeigen. Sie

können diese Zeichen sichtbar machen, indem Sie <CTRL>-<C> eingeben oder die RUN/STOP-Taste betätigen. Dadurch wird die Anzeige der Kontroll- Zeichen ein- oder ausgeschaltet. Wir wollen das nun erklären.

Leerzeichen oder andere Buchstaben:

"Zeichen" sind Buchstaben etc., die auf dem Bildschirm sichtbar sind. Daneben gibt es aber auch Zeichen wie Kontrol "M" und Kontrol"I", die wir gerade besprochen haben, die allerdings nicht sichtbar sind. Sogar Leerzeichen, die auf dem Bildschirm weiß erscheinen, sind in Wirklichkeit Buchstaben. Nun folgen die verschiedenen Buchstaben, die auf dem Bildschirm erscheinen können:

... Alphabetische Zeichen - sind "A" - "Z" als Groß- und Kleinbuchstaben. Nachdem MACROFIRE gebootet hat, befinden Sie sich im normalen Schreibmode mit Groß- und Kleinbuchstaben. Damit Sie nur in Großbuchstaben schreiben können, müssen Sie die <SHIFT/LOCK>-Taste drücken. Sie kehren in den normalen Schreibmode zurück, indem Sie erneut <SHIFT/LOCK> drücken. Passen Sie auf, wenn Sie die <C>-Taste oder <SHIFT>-<RETURN> drücken, und betätigen Sie nicht aus Versehen eine der Funktionstasten auf der rechten Seite der Tastatur.

. .. numerische Zeichen sind die Zahlen "0" - "9" in der obersten Zeile der Tastatur.

... Symbole - Dies sind die Zeichen über den Zahlen (z. B. "! ", "@") aber auch "+", "-", "=", "*" und andere nicht alpha-numerische Tasten.

... Control-Zeichen - werden erzeugt, indem die <CTRL>-Taste und gleichzeitig ein Buchstabe gedrückt werden. Ist ein Control-Zeichen darstellbar, so erscheint es als inverses Zeichen (Großbuchstabe).

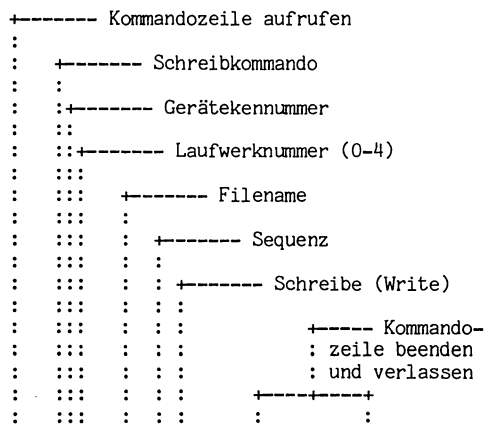
"filespec"

Dieses Wort bezeichnet den Namen eines Disketten- oder Cassettenfiles. Es bedeutet "File Specification" und

bezieht sich auf den Namen, den Sie dem abgespeicherten Text gegeben haben. Diese Files können entweder genau das beinhalten, was auf dem Bildschirm dargestellt wird, oder so formattiert sein, wie der Ausdruck auf dem Papier. Betrachten Sie solche Files als einzelne "Blätter Papier" auf der Diskette, worauf sich der Text befindet. Die Diskette bzw. Cassette ist das Speichermedium, auf dem die Files abgelegt sind.

Eine vollständige Beschreibung von Disketten- und Cassettenfiles entnehmen Sie bitte Ihrem C-64-Manual. Wir behandeln von jetzt ab nur noch die Informationen, die Sie speziell für Files beim MACROFIRE kennen müssen.

Der C-64 kann Ein- und Ausgabe-Operationen (bekannt als "I/O") mit verschiedenen Geräten durchführen. Z.B. mit der Tastatur, dem Bildschirm oder den Diskettenstationen. Jedes Gerät hat eine Nummer. Die Diskettenstationen haben z.B. #8. Wenn Sie mehr als ein Diskettenlaufwerk haben, können Sie zusätzlich eine Laufwerknummer mit in den Filenamen einbeziehen. Wenn Sie über die Kommandozeile ein Filespec eingeben wollen, müssen Sie ebenfals eine Gerätenummer mit angeben. Um ein Textfile auf Disk abzuspeichern, rufen Sie die Kommandozeile mit <CTRL>-<A> auf. Der Cursor muß sich am Anfang des Textes oder an der Stelle, von wo Sie speichern wollen, befinden.

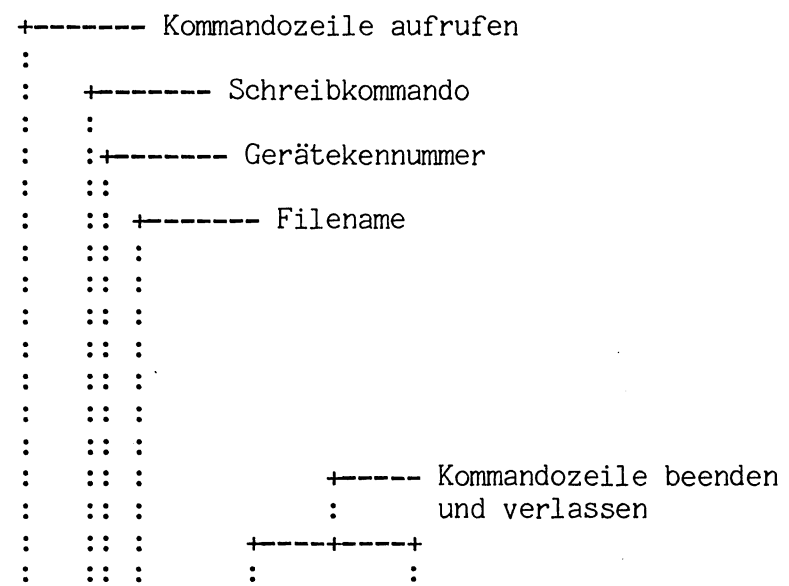


<CTRL>-<A>W80:NAME,S,W<CTRL>-<A><CTRL>-<A>

Wenn das File bereits vorhanden ist, können Sie es durch Einfügen eines @ nach der 8 überschreiben:

<CTRL>-<A>W8@0:NAME,S,W<CTRL>-<A><CTRL>-<A>

Um ein Textfile auf Cassette zu speichern, geben Sie ebenfalls <CTRL>-<A> ein, um die Kommandozeile aufzurufen.



<CTRL>-<A>W1NAME<CTRL>-<A><CTRL>-<A>

WICHTIGER HINWEIS

Wenn nichts mehr funktioniert und Sie glauben, daß das Programm abgestürzt ist: Drücken Sie die <RESTORE>-Taste und dann irgendeine Taste nach der Copyright-Meldung.

GERÄTEVORAUSSETZUNG

MACROFIRE wird auf Cassette oder Diskette geliefert

als Mindestausstattung benötigen Sie:

1) Commodore 64

2) Drucker (C-64-kompatibel!!)

3) Diskettenstation 1541

4) Commodore Datasette

ANWEISUNGEN ZUM LADEN

Cassettenversion

Schalten Sie den Computer und alle Peripheriegeräte aus. Legen Sie die MACROFIRE-Cassette in Ihre Datasette ein. Überzeugen Sie sich, daß das Band zurückgespult ist. Schalten Sie Ihren C-64 ein und laden Sie die Cassette mit

LOAD"MACROFIRE" <RETURN>

MACROFIRE wird nun geladen. Gestartet wird MACROFIRE durch RUN.

Diskettenversion

Schalten Sie den Computer aus und die Diskettenstation an. Warten Sie bis die "Busy"-Lampe erlischt, öffnen Sie dann den Diskettenschacht und schieben Sie die Diskette vorsichtig hinein (Aufkleber nach oben! !). Schließen Sie den Diskettenschacht und schalten Sie den Computer ein. Geben Sie LOAD "MACROFIRE*", 8 ein und starten Sie MACROFIRE durch RUN.

Der Editor

MACROFIRE meldet sich mit der Copyright-Meldung, nachdem er gestartet worden ist. Drücken Sie irgendeine Taste, um mit der Texteingabe zu beginnenen. Wenn der Editor Bildschirm erscheint, sehen Sie in der ersten Zeile verschiedene Buchstaben und Zahlen. Dies ist die sogenannte "Statuszeile". Sie gibt Informationen über den Status von MACROFIRE an. Von links nach rechts lesen Sie dort P:, T: und C:, jeweils von einer Zahl gefolgt.

P: Die Anzahl der Zeichen vor dem Cursor bis zum Textanfang wird hier angegeben.

T: An dieser Stelle wird angezeigt, wieviel Text Sie noch eingeben können, ohne die Speicherbegrenzung zu überschreiten. Die Zahl hängt davon ab, wieviel Text Sie bereits eingegeben haben, und welche Speicherkapazität Ihr Computer hat.

C: Zeigt an, wieviel Platz im "Kopierregister" frei ist. Es können auf einmal maximal 1024 (1K) Zeichen kopiert werden.

Auf der rechten Seite der Statuszeile steht das Wort OK. Das bedeutet, daß Sie sich in der normalen Arbeitsweise befinden. Diese Meldung ist eine von vielen "Zwei-Buchstaben-Meldungen", die an dieser Stelle des Bildschirms erscheinen können. Dieses "Meldefenster" warnt Sie vor Fehlern, zeigt an, ob das Kopierregister geöffnet ist und übernimmt noch verschiedene andere Aufgaben, die wir später behandeln wollen.

Unten am Bildschirm sehen Sie eine Zeile mit Punkten - die sogenannte "Kommandozeile". Diese Zeile wird herangezogen, um MACROFIRE besondere Anweisungen zu geben, die er ausführen soll, so z. B. Laden oder Abspeichern von Text.

Unter der Statuszeile sehen Sie oben links ein weißes Quadrat. (Diese Position ist die sogenannte "Home-Position). Das weiße Zeichen ist der "Cursor". Er zeigt an, wo das nächste Zeichen, welches eingegeben wird, auf dem Bildschirm erscheint.

Um mit der Texteingabe zu beginnen, tippen Sie einfach etwas ein. Bei der Eingabe wandert der Cursor von Links nach Rechts, bis er die 50. Zeichenposition auf dem Bildschirm erreicht. Wenn Sie mehr als 50 Zeichen eingeben, verschwindet der Cursor am rechtn Zeilenende. Um diese Zeichen bis hin zur 80. Position sehen zu können, geben Sie einfach <CTRL>-<V> ein. Dadurch wird die Zeile abgeschnitten. Um in den normalen Mode zurückzukehren, geben Sie wieder <CTRL>-<V> (ein/aus) ein.

Wenn Sie eine Taste für ca. 1/2 Sekunde drücken, setzt die Repeat-Funktion ein und zwar so lange, bis Sie die Taste wieder loslassen.

Tippen Sie nun z.B. das folgende kleine Programm ein:

```
            OUT LNM,3
            ORG $C000
            LDX #$00 ZAELER AUF NULL
START       LDA $0000,X
            JSR $AB47 AUSGABE EINES ZEICH.
            INX       INCREMENTIERE X REG.
            BNE START
            BRK
```

MACROFIRE bietet Ihnen eine Möglichkeit, die Textzeile so abzubrechen, daß sie günstig auf dem Bildschirm dargestellt werden können. Wenn Sie an den Punkt gelangen, wo Sie eine neue Zeile beginnen wollen, geben Sie einfach <RETURN> ein und der Cursor springt in die nächste Zeile.

Notizen

Die Befehle

Es gibt zwei Arten an Befehlen bei MACROFIRE:

"Kontroll-Kommandos", die sofort ausgeführt werden und "Anweisungen", die über die Kommandozeile ausgeführt werden. Wenn Sie erst einmal die verschiedenen Befehle gelernt haben, ist das hin und her schalten zwischen den beiden Arten zweitrangig.

Seien Sie sich immer klar darüber, in welcher Arbeitsweise Sie sich befinden, damit Sie nicht aus versehen Text löschen. Z.B. wird durch <CTRL>-<K> das Kopierregister gelöscht, aber die Eingabe von <CTRL>-<A> <K> <CTRL>-<A> <CTRL>-<A> (erscheint als: "K#") in der Kommandozeile löscht den ganzen Text im Speicher! !! Seien Sie deshalb vorsichtig bei Eingaben in der Kommandozeile und überprüfen Sie lieber alles noch einmal, bevor Sie sie ausführen.

KONTROLLKOMMANDO

Kontrollkommandos werden erzeugt, indem man zuerst die <CTRL>-Taste drückt und dann eine der unteren Tasten, wobei die <CTRL>-Taste gedrückt bleibt. Es gibt folgende Kontrollkommandos:

<CTRL>-<A>	Kommandozeile öffnen und schließen
<CTRL>-<D>	zeige Inhalt von Laufwerk 0
<CTRL>-<F>	schließe Kopierregister
<CTRL>-<G>	wiederhole Kommandozeile
<CTRL>-<H>	Anzeige der Kontrollzeichen (ein, aus)
<CTRL>-<I>	Cursor zur nächsten TAB-Position
<CTRL>-<J>	Inhalt des Kopierregisters an aktueller Cursorposition einfügen
<CTRL>-<K>	Kopierregister löschen
<CTRL>-<P>	Rufe Supermon 64 von Butterfield auf
<CTRL>-<Q>	das gleiche wie Cursor nach unten
<CTRL>-<R>	öffne Kopierregister
<CTRL>-<S>	Cursor an Textanfang das gleiche wie HOME
<CTRL>-<T>	das gleiche wie lösche rückwärts
<CTRL>-<V>	Schalte Zeilenumbruch ein/aus
<CTRL>-<X>	lösche nächste Zeile
<CTRL>-<Y>	Starte MACROFIRE Assembler
<CTRL>-<Z>	Endzeichen für MAROFIRE Assembler
<CTRL>-<^>	zeige Status von Laufwerk 0
<Cursor>-<UP>	Cursor an den Anfang der letzten Zeile
<Cursor>-<DOWN>	Cursor an den Anfang der nächsten Zeile
<Cursor>-<LINKS>	Cursor ein Zeichen zurück
<Cursor>-<RECHTS>	Cursor ein Zeichen vorwärts
<INST/DEL>	Cursor rückwärts löschen
<SHIFT>-<DEL>	ein Zeichen vorwärts löschen
<HOME>	Cursor an den Textanfang
<SHIFT>-<HOME>	Cursor an Textende

Die Kontrollkommandos können überall im Text eingegeben werden. Alle nicht genannten <CTRL>-Zeichen und sonstige werden an der aktuellen Cursorposition

eingefügt. Der Cursor ist dann um eine Stelle aufgerückt.

Im Folgenden finden Sie jetzt eine genaue Beschreibung der <CTRL>-Funktionen nach Funktionsgruppen geordnet:

Cursor-Steuerung (ohne Löschen)

Die Steuertasten, die wir jetzt besprechen, bewegen den Curser innerhalb des Textes, ohne etwas zu verändern. Wir wollen den Beispieltext, den wir gerade eingegeben haben, dazu benutzen, um zu zeigen, wie sich die Befehle auswirken.

<SHIFT>-<CLR/HOME> - bewegt den Cursor ans Textende

Dieser Befehl setzt den Cursor ans Ende des Textes, der sich im Speicher befindet. Wenn Sie diese Tasten betätigen, wird der Cursor am Ende des Textes auf dem Bildschirm erscheinen.

<HOME> oder <CTRL>-<S> - bewegt den Cursor an den Textanfang

Dieser Befehl setzt den Cursor vor das erste Zeichen im Text. Drücken Sie diese Tasten und Sie werden beobachten, daß der Cursor vor die erste Textzeile springt.

<CURSOR>-<Pfeil nach unten> oder <CTRL>-<Q> - bewegt den Cursor an den Anfang der nächsten Zeile

Dieser Befehl bewegt den Cursor an den Anfang der nächsten Zeile auf der linken Bildschirmseite. Wenn der Cursor sich auf der 16. Bildschirmzeile befindet, wird der Text nach oben gescrollt, um die nächste Zeile ausgeben zu können. Geben Sie diesen Befehl nun ein und

beobachten Sie, wie der Cursor sich Zeile für Zeile nach unten bewegt. Dabei wird die Zeile, in der sich der Cursor befindet, um einen Leerraum nach rechts verschoben.

<Cursor>-<Pfeil nach oben> - bewegt den Cursor an den Anfang der vorherigen Zeile

Dieser Befehl versetzt den Cursor an den Anfang der vorhergehenden Zeile auf der linken Bildschirmseite. Probieren Sie diesen Befehl im Beispieltext aus und beobachen Sie, wie der Cursor wieder nach oben wandert.

<Cursor>-<Pfeil nach links> - bewegt den Cursor ein Zeichen zurück

Dieser Befehl verschiebt den Cursor eine Position nach links. Wobei das Zeichen, welches sich auf dieser Position befindet, selbstverständlich um eine Position nach rechts versetzt wird.

<CURSOR>-<Pfeil nach rechts> - bewegt den Cursor eine Position nach vorne

Dieser Befehl bewegt den Cursor um eine Position nach rechts (oder nach unten zur nächsten Zeile, wenn er am Zeilenende ist) und verschiebt das Zeichen auf dieser Stelle nach links.

Cursor-Steuerung (mit Veränderung des Textes)

Die folgenden Befehle bewegen den Cursor, wobei der Text allerdings auf verschiedene Arten verändert wird.

<INST/DEL> oder <CTRL>-<T> - löscht das letzte Zeichen

Dieser Befehl löscht das Zeichen vor dem Cursor und bewegt den Text hinter dem Cursor um eine Position nach links.

<CTRL>-<I> - fügt TAB ein

Dieser Befehl bewegt den Cursor zur nächsten TAB-Position. Wenn sich irgend ein Text hinter dem Cursor befindet, wird er ebenfalls mit dem Cursor verschoben. In Wirklichkeit wird hier ein <CTRL>-<I> in den Text eingefügt. Wenn die Anzeige von Kontrollzeichen ermöglicht ist, (<CTRL>-<C>) erscheint ein inverses "I". Dabei wird der Cursor um eine Position nach rechts verschoben. Wenn die Anzeige für Kontrollzeichen ausgeschaltet ist, werden statt <CTRL>-<I> soviele Leerzeichen ausgegeben, wie durch die TAB-Funktion bestimmt worden sind.

<SHIFT>-<INSTL/DEL> löscht das nächste Zeichen

Dieser Befehl löscht das Zeichen rechts vom Cursor und verschiebt den Text nach links.

<CTRL>-<X> - löscht die letzte Zeile

Kontrollkommandos für das Kopierregister

Zuvor haben wir das Kopierregister benützt, um den Text, den wir für unsere Beispiele brauchen, vorübergehend festzuhalten . Nun wollen wir uns dieses Kopierregister einmal etwas genauer ansehen. Die folgenden Kontrollkommandos greifen auf den Kopierbuffer zurück. Wenn Sie irgend etwas in das Kopierregister übertragen, wird dabei der Text nicht verändert.

Kopierregister öffnen

<CTRL>-<R>

öffnet das Kopierregister. Setzen Sie den Cursor an das Ende des Textes, den Sie in das Kopierregister übertragen wollen, und drücken diese Tastenkombination.

Nun müssen Sie den Cursor an den Anfang des Textes, der kopiert werden soll, bewegen, (dabei können Sie jedes Cursorsteuerzeichen verwenden) und drücken dann <CTRL>-<F>, um den Puffer zu schließen.

Kopierregister schließen

<CTRL>-<F>

Schließt das Kopierregister für weitere Eingaben. Benützen Sie diesen Befehl, bevor Sie den Cursor an eine andere Stelle bewegen, nachdem Sie das Kopierregister gefüllt haben.

Inhalt des Kopierregisters in den Text einfügen

<CTRL>-<J>

Dieser Befehl fügt den Inhalt des Kopierregisters an der momentanen Stelle des Cursors ein. Bewegen Sie den Cursor einfach an die Stelle, wo Sie den Inhalt des Kopierregisters einfügen wollen, und drücken Sie die Tasten <CTRL>-<J>. Der Inhalt des Kopierregisters wird an dieser Stelle des Textes eingefügt, und alles, was sich hinter der momentanen Cursorposition befindet, wird ans Ende des eingefügten Textes verschoben.

Kopierregister löschen

<CTRL>-<K>

Mit diesem Befehl wird das Kopierregister gelöscht und der Zähler zurückgesetzt.

So arbeiten Sie mit dem Kopierregister: Bewegen Sie den Cursor ans Ende des Textes, der dupliziert werden soll. Öffnen Sie dann den Buffer mit <CTRL>-<R> und gehen Sie mit Hilfe der Cursorsteuerzeichen an den Anfang des zu kopierenden Textes (<Pfeil nach oben> (an den Anfang der letzten Zeile), <Pfeil nach rechts> (ein Zeichen zurück) oder <Home> (Cursor an Textanfang)). Danach wird das Kopierregister durch <CTRL>-<F> geschlossen.

Um Text an eine andere Stelle verschiebenzu können, müssen Sie den Originaltext später löschen. Sie können dazu das Kopierregister füllen und dabei den Originaltext löschen, indem Sie <INST/DEL> (letztes Zeichen löschen) und <CTRL>-<X> (letzte Zeile löschen) verwenden.

Andere Kontrollzeichen Druckersteuertaste

Das Drücken der Taste <STOP> während der Ausgabe auf einen Drucker beendet die Ausgabe.

Steuerzeichen für die Bildschirmausgabe

<CTRL>-<C>

Ein-/ausschalten der Steuerzeichendarstellung. Wenn die Anzeige der Kontrollzeichen eingeschaltet ist, wird jedes TAB und <RETURN> angezeigt. Ist es aber ausgeschaltet, werden für ein TAB entsprechend viele Leerzeichen auf dem Bildschirm ausgegeben und <RETURN> ist unsichtbar.

Kontrollkommandos für die Diskette:

<CTRL>-<D>

Durch Tippen dieser Tasten wird Ihnen der Inhalt von Diskettenlaufwerk 0 angezeigt. Sie kehren in den Editor zurück, indem Sie irgendeine Taste drücken.

<CTRL>-<^>

Gibt Ihnen den Fehlerstatus der Diskette aus. Angezeigt wird Fehlernummer, Fehlermeldung, Spurnummer und Sektornummer. Drücken Sie irgendeine Taste, um zum Editor zurückzukehren.

Steuerzeichen für den Assembler und Maschinen-Monitor

<CTRL>-<Y>

Starten des Assemblers. Der Assembler übersetzt den Code im Textpuffer vom Anfang bis zum Ende des Textes oder bis zu einem <CTRL>-<Z>.

<CTRL>-<P>

Aufruf des eingebauten Monitors (Supermon von J. Butterfield). Die Kommandos entnehmen Sie bitte der Beschreibung für Supermon. Achtung: durch das X-Kommando gelangen Sie wieder über die Copyright-Meldung (durch das Drücken einer Taste) in den Editor. Es erfolgt kein Sprung in BASIC!

Steuerkommandos für die Kommandozeile

Die beiden folgenden Befehle werden dazu benützt, die Kommandozeile aufzurufen oder auszuführen.

<CTRL>-<A> - Kommandozeile öffnen/schließen

<CTRL>-<A> - öffnet, schließt und führt die Kommandozeile aus, wie wir bald sehen werden.

<CTRL>-<G> - Kommandozeile ausführen

Dieser Befehl führt die zuletzt eingegebene Kommandozeile aus. Dies ist nützlich, um eine Kommandozeile zu wiederholen die durch einen aufgetretenen Fehler abgebrochen, wurde und man den Fehler ausgebessert hat.

Die Kommandozeile

Um in der Kommandozeile etwas eingeben zu können, müssen Sie zuerst <CTRL>-<A> drücken. Dadurch wird der erste Punkt der Kommandozeile durch ein Dollarzeichen ("$") ersetzt. Sie können dann beliebig viele Kommandos eingeben, die Sie jeweils durch <CTRL>-<A> (durch "$" angezeigt) trennen. Um die Kommandozeile auszuführen, drücken Sie einfach <CTRL>-<A>, <CTRL>-<A>. Es wird dann am Ende der Zeile ein Dollarzeichen, und wenn die Kommandos ausgeführt worden sind, noch ein "Nummerzeichen" ("#") ausgegeben.

Durch die Kommandozeile können Sie z. B formatierten Text ausdrucken lassen, Files von der Diskette laden, verschiedene Editorfunktionen ausführen und die I/O-Kommandos der Commodore-Diskette benützen.

Folgende Kommandos sind durch die Kommandozeile ansprechbar:

<@>	Setze die Tabulatorbreite
<B>	Ein Zeichen zurück
<C>	Change (Suchen und Ändern)
<D>	Lösche letztes Zeichen
<F>	Ein Zeichen vorwärts
<G>	Füge Kopierregister ein
<H>	Füge Hexbyte ein
<I>	Füge String ein
<J>	Springt zum Kommandozeilenanfang
<K>	Lösche Textbuffer!!!
<L>	Gebe formattierten Text auf ein Gerät aus (Drucker oder Bildschirm oder sogar Disk)
<M>	Gebe formattierten Text auf einen Commodoredrucker aus
<P>	Senden Kommando(s) an Diskette
<R>	Lese File ein
<S>	Suche einen String
<T>	Lösche nächste Zeile
<U>	Rufe Userprogramm auf
<W>	Schreibe File

Manche Kommandos können mehrfach hintereinander ausgeführt werden. Man gibt eine Zahl von 1 - 255 vor dem Kommando ein. Auch läßt sich die ganze Zeile wiederholen, durch das "J"-Kommando (Jump). Es wird jetzt wieder an den Anfang der Zeile zurückgesprungen. Wichtig ist jetzt natürlich, daß es einen Abbruch geben wird, z.B ein String, der im ganzen Text ersetzt werden muß, wird nicht mehr gefunden. Das Programm steigt dann mit einer Fehlermeldung sofort aus und man ist wieder im Kontrollmode.

Unten finden Sie nun nähere Erläuterungen der Kommandos, die in der Kommandozeile benützt werden können:

<@> setze eine TAB Position
Das Format: @<n> <CTRL>-<A> (n=1-9)
Z.B.: $@5$#

Dieser Befehl setzt für ein TAB jedesmal fünf Leerräume. Durch Drücken von <CTRL>-<I> springt der Cursor jedesmal zur nächsten TAB Position. Wenn die Anzeige von Kontrollzeichen ausgeschaltet ist, (<CTRL>-<C>) sehen Sie die Anzahl der Leerzeichen bei jedem TAB. Ist die Kontrollzeichendarstellung eingeschaltet, sehen Sie bei jedem TAB ein inverses "I", und der Cursor erscheint nur eine Position weitergerückt. Während dem Ausdruck werden TABs ignoriert, wenn Formattieranweisungen eingegeben worden sind. In einem unformatierten Ausdruck erscheinen TABs wie im Text, wenn die Darstellung der Kontrollzeichen ausgeschaltet ist. TABs werden in Textfiles als Kontrollzeichen gespeichert und belegen somit weniger Platz, als eine entsprechende Anzahl Leerzeichen. TAB-Positionen sind immer Vielfache von "n".

<H> füge Hex-Byte ein
Format: H<Byte>ESC
z.B.: $H1D$#

Dadurch wird die Hex-Zahl $1D an der momentanen Cursorposition eingesetzt (der Wert $1D ist das Zeichen für EOF bei einem Disketten/Cassetten-File). Der Befehl kann bis zu 255 mal wiederholt werden, indem Sie die entsprechende Zahl vor dem "H" eingeben. Z.B. $25H20$#.

<I> füge Zeichenkette ein.
Format: I<String>ESC
z.B.: $IHallo$#

Dieser Befehl fügt das Wort "Hallo" an der aktuellen Cursorposition ein. Wiederholtes Einfügen kann durch Eingeben der gewünschten Anzahl vor "I" erfolgen.

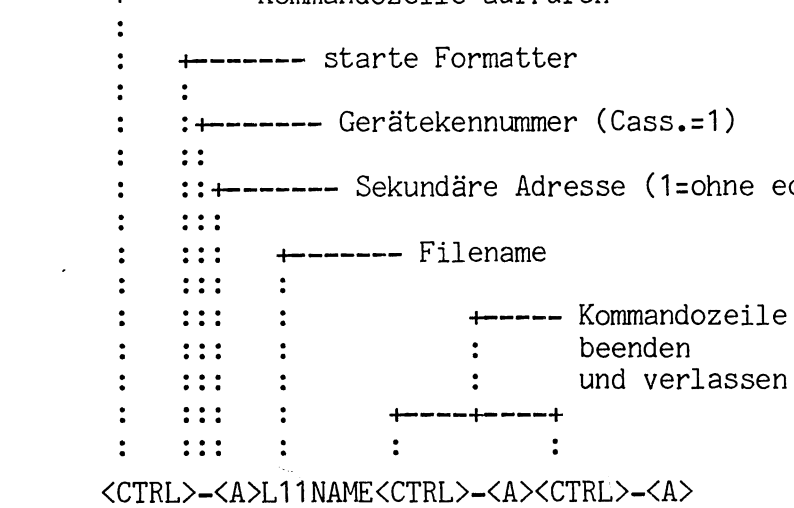
<J>

springe an den Anfang der Kommandozeile. Dieser Befehl kann sehr nützlich für Sie sein, wenn Sie im Text irgendetwas löschen oder ändern wollen, aber nicht wissen, wie oft es geändert werden muß. Mit dem Befehl springen Sie einfach an den Anfang der Kommandozeile und führen Sie abermals aus. Das Kommando "J" wird so lange ausgeführt, bis ein Fehler auftritt oder alle Buchstaben oder Worte verändert worden sind. Sie brauchen nicht <CTRL>-<G> zu drücken. Setzen Sie den Befehl nicht ein, wenn kein Fehler auftreten kann. Z.B. werden durch $STEXT$4ditext$J$# alle "TEXT" in "text" die sich im Textbuffer befinden, geändert.

<L>

Sie können mit diesem Befehl ein File an jedes Gerät ausgeben, außer an die Tastatur! ! Dabei wird der normale ASCII-Code ausgegeben, einschließlich Carriage Return und Zeilenrücklauf. Mit diesem Befehl wird der Text auch auf Ihren Drucker ausgegeben. Der Befehl arbeitet nicht mit Ihrem Commodore-Drucker. Wenn Sie einen Commodore-Drucker haben, benützen Sie bitte den "M-Befehl".

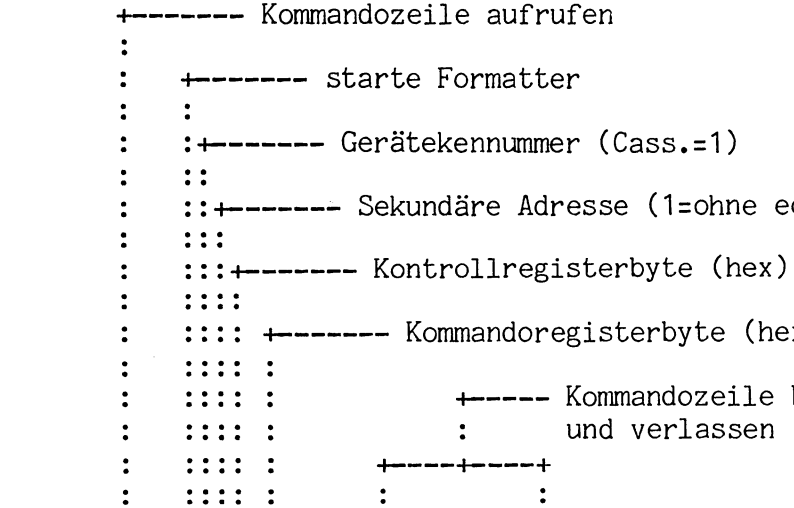
1) Listen auf eine Cassette



<CTRL>-<A>L11NAME<CTRL>-<A><CTRL>-<A>

2. Über RS232 listen

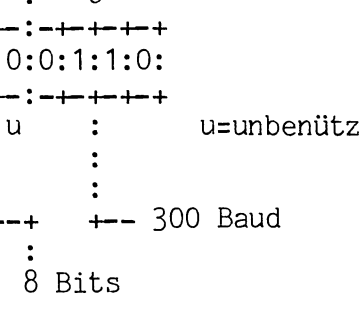
Den Anschluß entnehmen Sie bitte Ihrem Referenz-Manual



<CTRL>-<A>L208610<CTRL>-<A><CTRL>-<A>

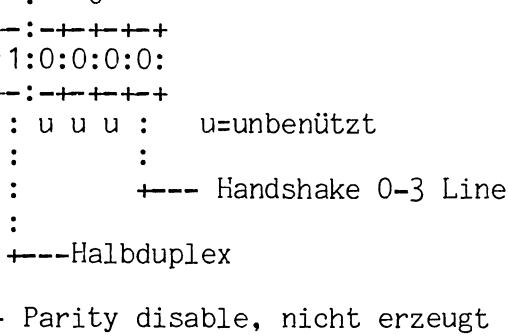
(*) Durch obiges Beispiel arbeitet das RS232 Interface mit folgender Einstellung:

a) Kontrollregisterbyte



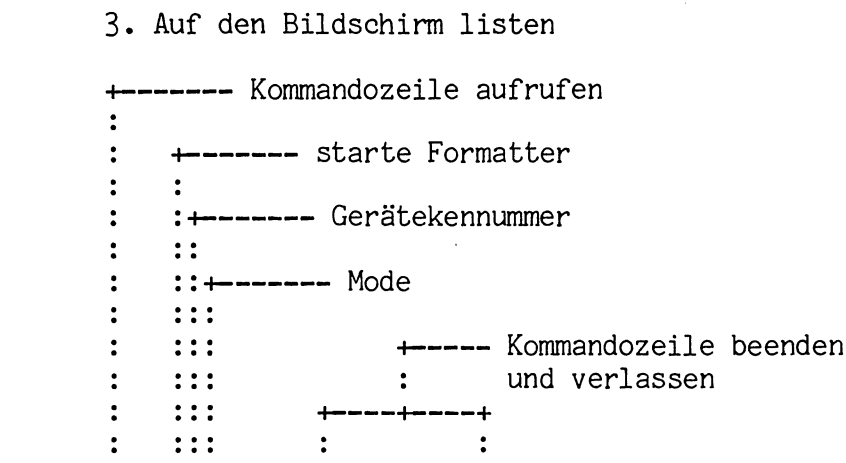
Siehe auch Commodore Programmer's Reference Guide auf Seite 350

b) Kommandoregisterbyte



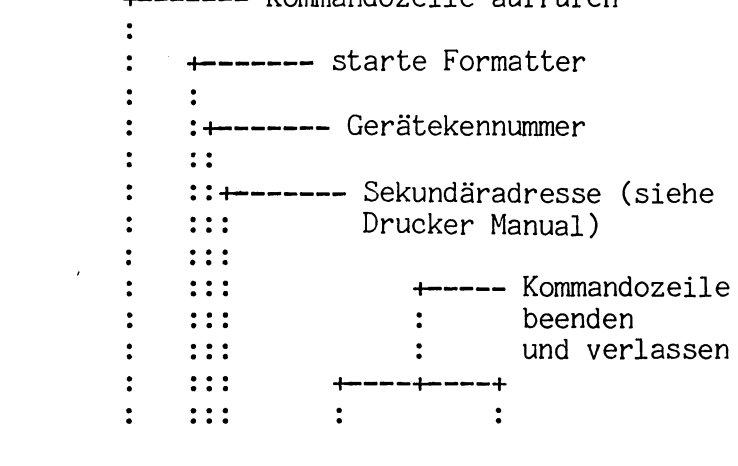
Wenn Sie die Spezifikationen der RS232 Schnittstelle ändern wollen, ziehen Sie bitte Ihr Handbuch zu Rate. Das Programm arbeitet normalerweise auf jedem RS232-Drucker. Wir haben es an einem Qume SPRINT 9 Drucker und einem Decwriter getestet. Im Manual ist nicht erwähnt, daß Sie die Anschlüssen wie z. B. Transmit Data usw. anschließen müssen.

3. Auf den Bildschirm listen



<CTRL>-<A>L30<CTRL>-<A><CTRL>-<A>

4. Ausgabe auf einen IEEE-Drucker über den IEEE-Bus. Arbeitet nicht mit Commodore Druckern bzw. kompatiblen Druckern. (Siehe "M-Befehl")

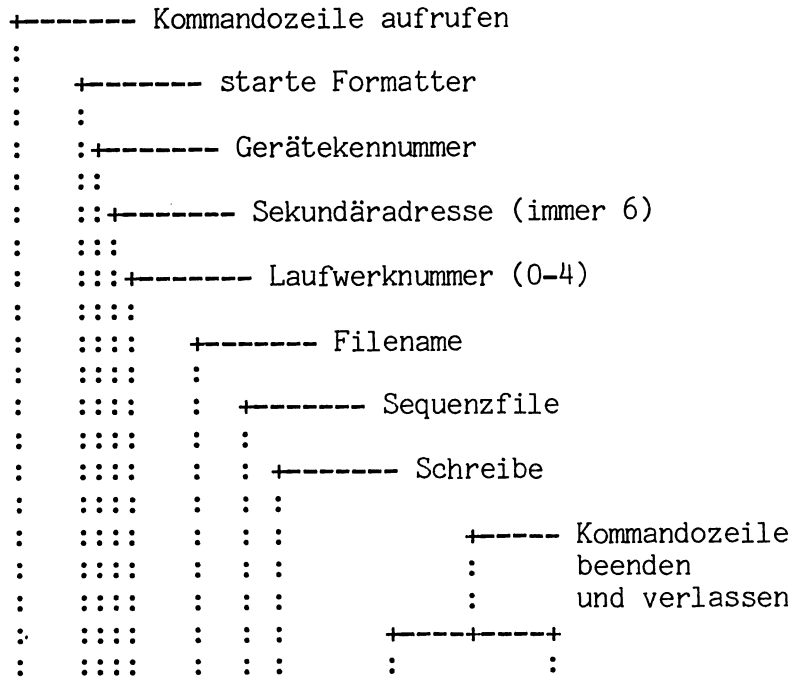


<CTRL>-<A>L40<CTRL>-<A><CTRL>-<A>

5. Auf Diskette listen

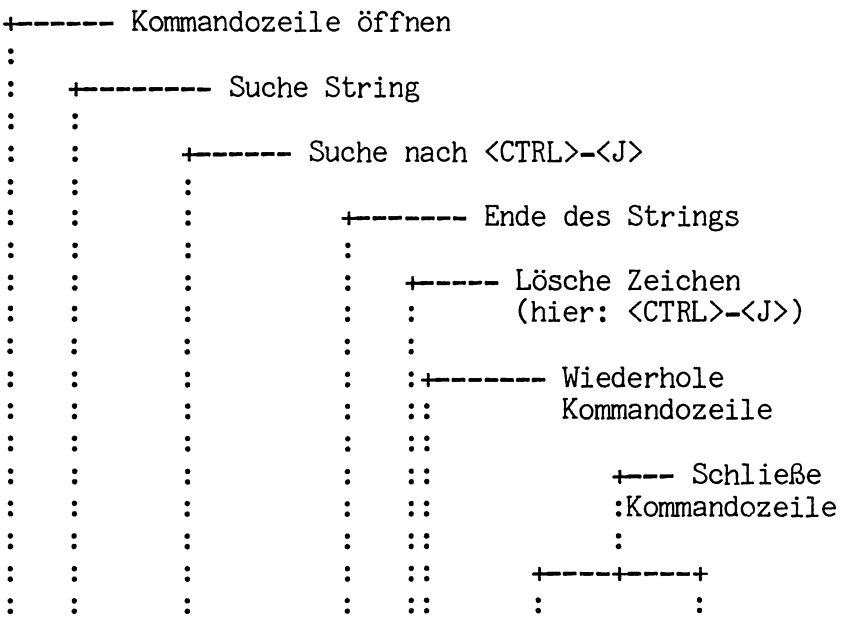
Eine besondere Eigenschaft von MACROFIRE ist, daß Sie formattierten Text auf die Diskette schreiben und später wieder in den Speicher einlesen können.

z.B.:



<CTRL>-<A>L860:NAME,S,W<CTRL>-<A><CTRL>-<A>

Durch die Kommandozeile von oben wird der Text, wie beim Drucker formatiert, auf die Diskette ausgegeben (einschließlich Zeilenvorschub und Wagenrücklauf. Der Text kann durch den READ-Befehl (wird später beschrieben) wieder in den Speicher gelesen werden. Die Zeilenvorschübe (<CTRL>-<J> sind dann im Text sichtbar und können durch folgende Eingabe entfernt werden:



<CTRL>-<A>S<CTRL>-<J><CTRL>-<A>DJ<CTRL>-<A><CTRL>-<A>

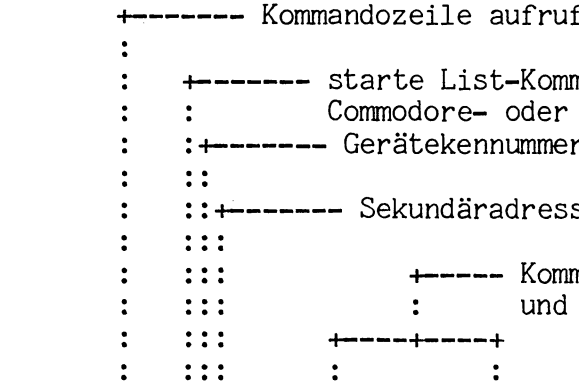
Achtung: Verwenden Sie als Sekundäradresse immer 6

<M>

gebe Text im Commodore-Format aus.
Der Befehl wird genau wie der "L" Befehl verwendet, aber es wird kein Zeilenvorschub gesendet.

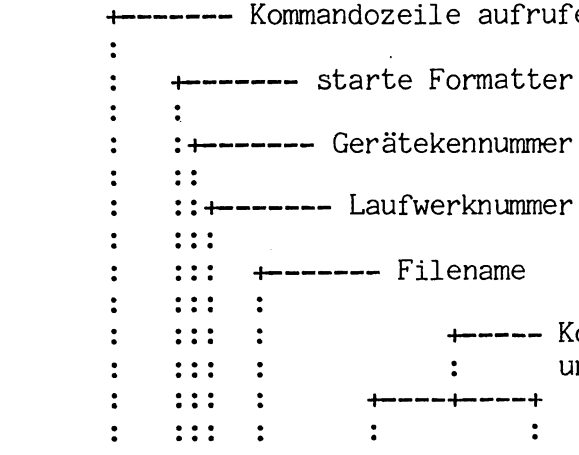
Benützen Sie den Befehl beim CBM 2022 und VIC 1525.

z.B. Ausgabe auf einen 2022 und VIC1525.



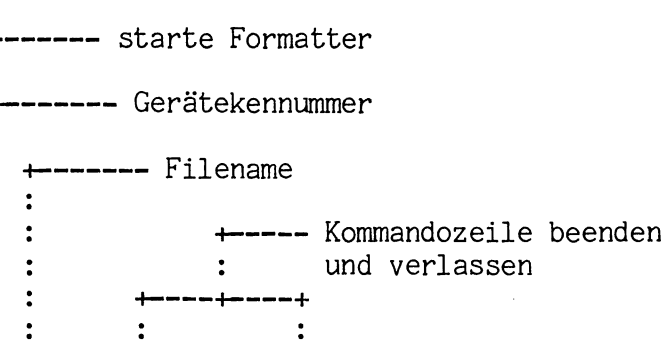
<CTRL>-<A>M40<CTRL>-<A><CTRL>-<A>

<R>
File einlesen. Das File wird an der aktuellen Cursorposition eingefügt. Nach dem Laden wird der Cursor ans Ende des gerade gelesenen Textes gesetzt. Wenn irgend ein Text vor dem Laden dem Cursor folgte, so ist er nachher auch noch an der gleichen Stelle. Text vor dem Cursor verbleibt im Speicher. Wenn Sie den geladenen Text allein im Speicher haben wollen, müssen Sie den anderen vorher löschen. Das Einlesen eines Textfiles von Diskette



<CTRL>-<A>R80:NAME<CTRL>-<A><CTRL>-<A>

Wie wird ein Textfile von Cassette eingelesen



<CTRL>-<A>R1NAME<CTRL>-<A><CTRL>-<A>

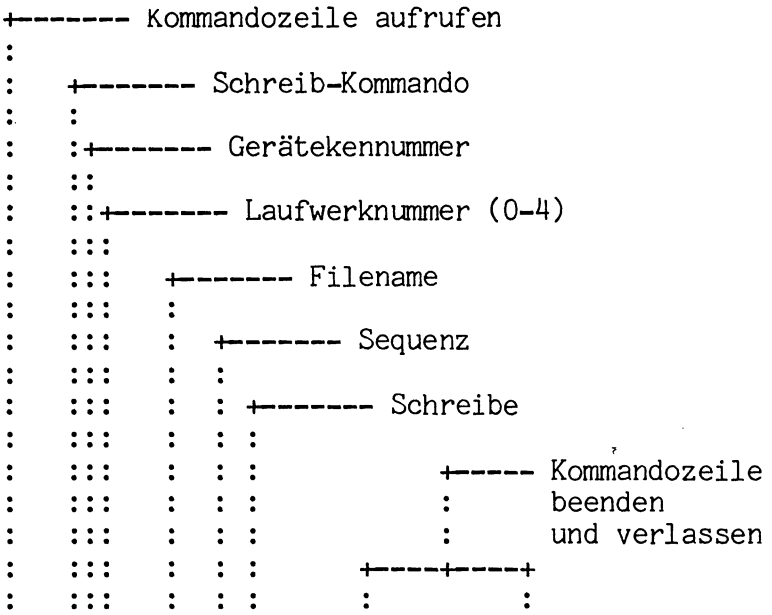
<S> Suche eine Zeichenkette
Format: S<string><CTRL>-<A>
z.B: $Sbaseball$#

Der Befehl sucht nach der gewünschten Zeichenkette und bleibt dort mit dem Cursor stehen. Wenn Sie eine Zahl "n" (1 - 255) vor dem "S" eingeben, so bleibt der Cursor an der n-ten Zeichenkette stehen. Z.B: $5Sund$# bewirkt, daß der Cursor am fünften "und" stehen bleibt. Sie können aber auch das Suchen manuell wiederholen, indem Sie jeweils <CTRL>-<G> und keine Zahl vor dem "S" eingeben. Es wird solange gesucht, bis das letzte Wort im Text erreicht worden ist. Wenn keine entsprechende Zeichenkette gefunden wird gibt der Computer auf dem Bildschirm oben rechts ein "S?" aus.

<U> Springe in Userprogramm ab Adresse $B000 (hexadezimal). Sie können Ihre Programme an dieser Stelle des Speichers ($B000 - $CFFF) ablegen.

Achtung: I/O-Operationen mit Druckern, Cassette oder Diskette benützen diesen Speicher ebenfalls!

<W> Um ein Textfile auf der Diskette zu speichern, geben Sie zuerst <CTRL>-<A> ein, um die Kommandozeile aufzurufen. Der Cursor muß dabei am Textanfang sein.

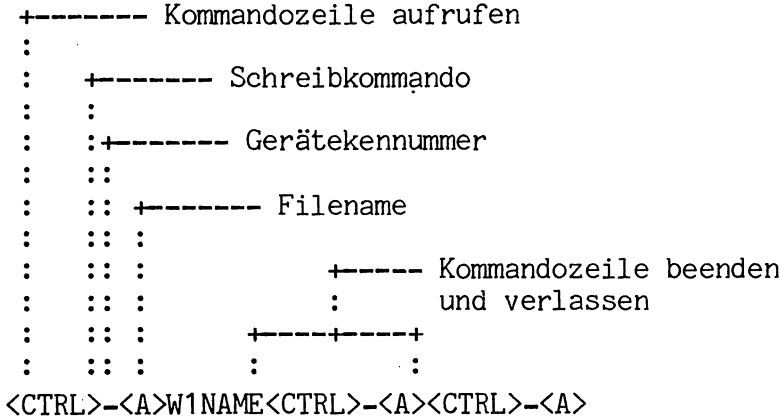


<CTRL>-<A>W80:NAME,S,W<CTRL>-<A><CTRL>-<A>

Sie können ein bereits vorhandenes File überschreiben, indem Sie nach der Zahl "8" ein "@" einfügen.

<CTRL>-<A>W8@0:NAME,S,W<CTRL>-<A><CTRL>-<A>

Um ein Textfile auf Cassette abspeichern zu können, geben Sie ebenfalls <CTRL>-<A> ein, um in die Kommandozeile zu gelangen.



<CTRL>-<A>W1NAME<CTRL>-<A><CTRL>-<A>

Beim Speichern wird der Bildschirm gelöscht. Sobald der Text fertig abgespeichert oder ein Fehler aufgetreten ist, springt MACROFIRE in den Editor zurück.

Fehlermeldung des Editors

Die Fehlermeldungen des MACROFIRE sind zwei Zeichen lang und werden auf dem Bildschirm oben rechts angezeigt. Wenn ein Fehler auftritt, wird die Meldung ausgegeben, und der Editor übernimmt wieder die Kontrolle. Hier nun die Fehlermeldungen mit ihren Erklärungen:

CO Comandline Overflow. Mehr als 40 Zeichen in der Kommandozeile.
E? irgendein ungültiges Kommando in der Kommandozeile!!
#? die Zahl vor einem Kommando ist größer als 255
I? kein Platz mehr im Textbuffer
C? Kopierregister ist voll
H? der Hexwert des H-Kommandos ist fehlerhaft (der Gültigkeitsbereich $00 - $FF)
T? der Tabulatorwert ist größer als 9 oder kleiner als 1
S? der gesuchte String wurde nicht gefunden
L? im L-Befehl stimmt etwas nicht (falsches File, während des Druckens gestopt usw.)
RW Lese/Schreibfehler (das File existiert nicht oder es ist ein Fehler beim lesen oder schreiben aufgetreten)
V? kein Inhaltsverzeichnis (beim Versuch, das Inhaltsverzeichnis zu lesen ist ein Fehler aufgetreten)

Wie wird der C 64 an einen seriellen Drucker angeschlossen?

Der Commodore 64 hat ein eingebautes RS232 Interface, um ihn an einem Drucker, einem Modem oder einem anderen Computer mit RS232 Interface, betreiben zu können.

Das RS232 Interface, das über den User Port arbeitet, ist keine richtige RS232 Schnittstelle mit einem Spannungswert von +-12 Volt. Sie haben nur einen TTL-Wert zur Datenübermittlung zur Verfügung. Wir haben dieses RS232 Interface mit TTL-Wert an einigen Druckern ausprobiert (Decwriter, Qume Sprint 9, Brother HR15 und Nec Spinwriter). Das RS232 Interface mit TTL-Wert arbeitet mit diesen Drucken und sogar mit dem Smart-Modem von Hayes einwandfrei.

Die Software für das RS232 Interface ist bereits vorhanden und die Spezifikationen für die Übertragung können durch das Setzen einzelner Register bestimmt werden.

Um die RS232 Drucker betreiben zu können, brauchen Sie folgende Teile:

1. Einen Stecker für den User Port mit 24 Pins
   (TRW Cinch 251-12-50-170/50-24sn-98124)
   Sie erhalten ihn bei Ihrem Computerhändler.
2. Einen RS232 Stecker, 25 Pins.

3. Zwei oder drei Drähte

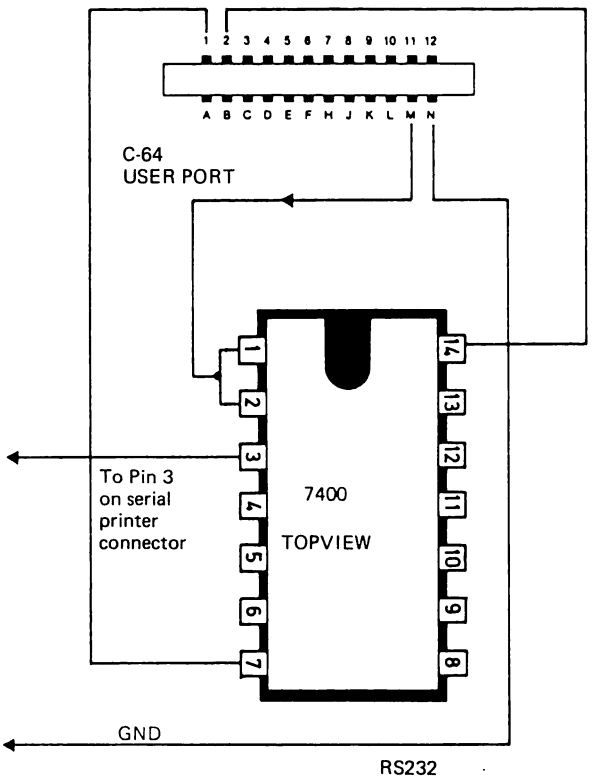
Es stellte sich heraus, daß ein Dreiweg-Interface ohne Handshaking auf einfachste Weise mit dem C-64 und einem Qume Sprint 9 Drucker anzuschließen ist. Um ein Dreiweg Interface anzuschließen, müssen Sie folgende Verbindungen herstellen.

Pin	Beschreibung	ein/aus	6526 Pin
C	RECEIVE DATA	EIN	PB0
B	RECEIVE DATA	EIN	FLAG2
M	TRANSMIT DATA	AUS	PA2
N	GROUND	-	GND

Wenn Sie einen RS232 Drucker anschließen wollen, brauchen Sie nur die beiden Leitungen GND und TRANSMIT DATA anschließen.

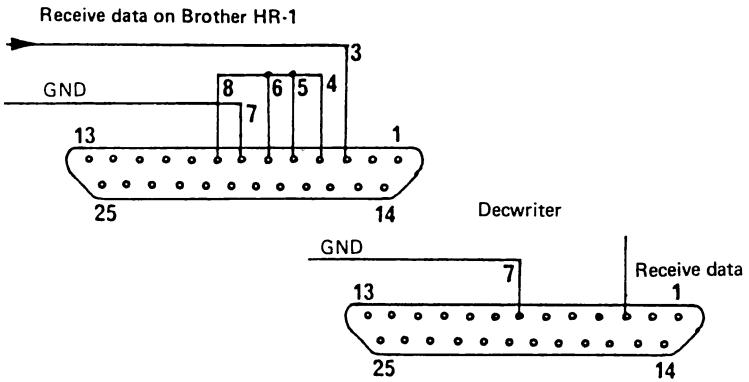
User-Port-Anschluß am C-64

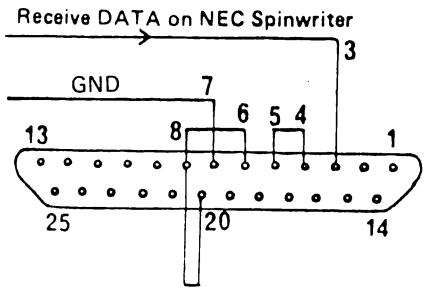
Wir haben herausgefunden, daß die Leitung TRANSMIT DATA (Pin M) invertiert werden muß, bevor Sie an der Drucker weitergeleitet werden kann. Dazu können Sie ein 7400 NAND-Gatter direkt an den User Port Stecker anschließen. Pin 1 und Pin 2 dienen der Stromzufuhr.



Wenn Sie mehr als 4 Leitungen invertieren wollen, benützen Sie einen 7404 Inverter Chip mit 6 Invertern.

Viele Drucker mit RS232 Schnittstelle brauchen eine Handshaking-Leitung. In diesem Fall müssen Sie Ihren Drucker folgenden Skizzen entsprechend anschließen: Wir haben es an drei Druckern getestet:





Der Assembler

Der Assembler wird vom Editor aus durch <CTRL>-<Y> gestartet. Der Assembler übersetzt den Quelltext, der im Textpuffer des Editors abgelegt ist, vom Textanfang bis zu einem <CTRL>-<Z> (Stopzeichen für Assembler) in drei Durchgängen. Wenn Sie kein <CTRL>-<Z> gesetzt haben, wird der Text bis zum Ende übersetzt. Beim zweiten Durchgang des Assemblierens wird auf der letzten Position in der obersten Zeile ein Zeichen gesetzt, welches anzeigt, daß der Assembler noch arbeitet. Im dritten Durchgang erfolgt dann die Ausgabe. Sie können den assemblierten Text auf den Bildschirm ausgeben, eine Liste der Marken anfertigen oder beides drucken lassen. Die Ausgabe kann auf den Bildschirm oder einen Drucker erfolgen.

Wenn beim Übersetzen keine Fehler aufgetreten sind, beendet der Assembler seine Arbeit regulär. Sie können dann durch Drücken einer Taste über die Copyright-Meldung in den Editor zurückkehren. (Um von der Copyright-Meldung in den Editor zu gelangen, ist selbstverständlich ein weiterer Tastendruck erforderlich).

Wird beim Assemblieren ein Fehler entdeckt, so wird sofort gestoppt und eine selbsterklärende Fehlerbeschreibung auf den Bildschirm ausgegeben. Drücken Sie dann eine Taste, um wieder (ebenfalls über die Copyright-Meldung) in den Editor zurückzukehren. Der Cursor steht dann hinter dem Textteil, durch den der Fehler hervorgerufen worden ist. Es ist dadurch möglich, den Fehler schnell und einfach zu beheben.

Format und Syntax der OP-Codes

Der Assembler erkennt alle OP-Codes des 6500-Befehlssatzes und noch einige zusätzliche Pseudo-OP-Codes, die zur Steuerung des Assemblers eingesetzt werden.

Die 6500-OP-Codes:

```
ADC AND ASL BCC BCS BEQ BIT BMI BNE BPL BRK
BVC CLC CLD CLI CLV CMP CPX CPY DEC DEX DEY
EOR INC INX INY JMP JSR LDA LDX LDY LSR NOP
ORA PHA PHP PLA PLP ROL ROR RTI RTS SBC SEC
SED SEI STA STX STY TAX TAY TSX TXA TXS TYA
```

Pseudo-OP-Codes:

EQU : EQUal wird benützt, um Marken zu definieren. Die Marke bekommt den Wert, der durch EQU definiert wird.

EPZ : Equal Page Zero. Ähnlich wie EQU, nur für Adressen der Seite Null.

ORG : ORiGine. Wird benützt, um die Anfangsadresse des assemblierten Object-Codes festzulegen. (Die Adresse, wo das Programm abgelegt werden soll.)

DFB : DeFine Byte. Einfügen des(r) folgenden Bytes.

DFW : DeFine Word. Wie DFB, nur für Worte. D.h. zwei Bytes pro Ausdruck (niederwertiges und höherwertiges Byte).

ASC : ASCII-String. Einfügen eines ASCII-Strings.

OUT : OUTput (siehe später).

INCLUDE : INCLUDE Sourcefile during assembly. Nach der Assemblierung befindet sich kein zusätzlich geladener Text im Puffer. Mit diesem OP-Code haben Sie die Möglichkeit, größere Sourcefiles zu assemblieren, als im Textpuffer Platz haben. (Assemblierung von Diskette)

MACRO : Anfang eines MACROs mit formaler Parameterliste (siehe später).
MEND : Ende eines MACROs.

Format der Adressierungen:

```
Impl.Acc : [label]_OPC[_;<com>]<RETURN>
Immediate: [label]_OPC_#<expr>[_<com>]<RETURN>
Abs,Zp,Re: [label]_OPC_<expr>[_<com>]<RETURN>
AbsX;ZpX : [label]_OPC_<expr>,X[_<com>]<RETURN>
AbsY;ZpY : [label]_OPC_<expr>,Y[_<com>]<RETURN>
IndX.    : [label]_OPC_(<expr>,X)[_<com>]<RETURN>
IndY.    : [label]_OPC_(<expr>),Y[_<com>]<RETURN>
Ind      : [label]_OPC_(<expr>)[_<com>]<RETURN>
```

Format der Pseudo-OP-Codes:

```
EQU      : <label>_EQU_<expr>[_<com>]<RETURN>
EPZ      : <label>_EPZ_<expr>[_<com>]<RETURN>
ORG      : [label]_ORG_<PL>[,PP][_<com>]<RETURN>
DFB      : [label]_DFB_<expr>[,<expr>]"[_<com>]<RETURN>
DFW      : [label]_DFW_<expr>[,<expr>]"[_<com>]<RETURN>
ASC      : [label]_ASC_<delim>[string]<delim>[_<com>]<RETURN>
OUT      : [label]_OUT_[L][N][M][C][,<dev>[,<sec>[,"<name>"]]]<RETURN>
INCLUDE  : [label]_INCLUDE_<dev>,<sec>,"<name>"<RETURN>
MACRO    : <label>_MACRO_[<formp>[,<formp>]"]<RETURN>
MEND     : [label]_MEND_[_<com>]<RETURN>
```

Legende zur Syntax:

Ausdrücke zwischen <> sind unbedingt notwendig. Ausdrücke innerhalb [ ] können weggelassen werden. Ein " nach] bedeutet, daß der ganze Ausdruck innerhalb der [] beliebig oft wiederholt werden kann. Ein _ bedeutet, daß mindestens ein Leerraum oder ein <CTRL>-<I> eingefügt werden muß. OPC stellt einen legalen OP-Code dar.

Textzeilen, die mit einem <*> oder <CTRL>-<L> beginnen, werden als Kommentar interpretiert und vom Assembler übersprungen. Eine Leerzeile (eine Zeile, die nur ein <RETURN> enthält, wird ebenfalls übersprungen.

Marke:

Besteht aus einem Buchstaben, der von Buchstaben oder Zeichen gefolgt wird. Jedes Zeichen ist signifikant.

String:

Zeichenkette aus ASCII-Zeichen. Die entsprechenden Werte werden an dieser Stelle gemäß ihrer Position in der Zeichenkette eingefügt. Ein String kann aus maximal 250 Zeichen bestehen und wird durch den selben Begrenzer <Delim> eingeleitet und beendet.

Delim:

Jedes nicht alfanumerische Zeichen. Bei Verwendung von Ö als Begrenzer wird automatisch 128 zum letzten Zeichen hinzuaddiert. (Signbit ein).

Com:

An dieser Stelle steht Kommentar.

<RETURN>:

die <Return>-Taste auf der rechten Seite ihrer Tastatur.

Ausdr:

Ein Ausdruck kann eine dezimale Zahl, eine hexadezimale Zahl (eingeleitet durch $ wie z.B. $40 oder $FFFB), eine Binärzahl (eingeleitet durch % wie z.B. %1011 oder %10101111), eine Marke, oder ein

ASCII- Zeichen (eingeleitet durch ' wie z.B. 'A oder '#) sein. Auch die Kombination mit arithmetischen Operationen (+, -, *, /) ist möglich. Ein Asterisk (*) als Term in einem Ausdruck wird als der momentane Wert des Programmzählers interpretiert. Dadurch ist relative Indexierung sehr einfach. Die Ausdrücke werden von links nach rechts aufgelöst und können mit Hilfe von Klammern () geschachtelt werden.

PL,PP:

PL = logische Startadresse des assemblierten Programms; d. h. das assemblierte Programm arbeitet nur, wenn es bei dieser Adresse anfängt.
PP = physikalische Start- (Dump-) Adresse des Programms; d. h. das Programm wird zwar ab dieser Adresse abgelegt, ist dort aber nicht lauffähig (außer PL = PP). Auf diese Art können Sie ein Programm assemblieren, obwohl es in einem belegten Speicherbereich liegen würde. Dieser Speicherbereich kann z.B. vom Textpuffer oder Tabellen belegt sein. Wenn Sie PP nicht definieren, so ist es automatisch gleich PL.

Ger:

Gerätenummer. Diese Zahl entspricht der Gerätenummer, die Sie dem entsprechenden Manual entnehmen können. Die Gerätenummer kann ein Ausdruck wie jeder andere sein.

Sek:

gibt eine Sekundär-Adresse an und ist im allgemeinen geräteabhängig. (Siehe entsprechende Manuals).

Achtung: bei Benützung der Diskette und der Kombination von INCLUDE und OUTput müssen die Sekundär-Adressen, die durch beide Pseudo-OP-Codes definiert werden, verschieden sein. Z.B.:

```
INCLUDE 8,5,"0:FILE"
```

arbeitet nur einwandfrei, wenn OUT folgendermaßen aussieht:

```
OUT LN,8,6,"0:OUTFILE,S,W"
```

Wie Sie sehen, ist die Sekundär-Adresse einmal 5 und beim anderen Mal 6, so daß keine Schwierigkeiten auftreten können.

Name:

der File-Name für I/O-Verkehr mit einem Gerät. Wenn Sie eine Diskette benützen, muß der Name ",S,W" beinhalten, wie Sie im Beispiel oben sehen können.

Formp:

der formale Parameter, der im Macro-Body benützt wird. (Zwischen MACRO und MEND). Der formale Parameter wird durch einen aktuellen Parameter im Macro-Body ersetzt, wenn der Macro aufgerufen wird. Z. B. ein Macro-Body, der irgendwo definiert worden ist:

```
SAMPLE MACRO A,B  (A und B als formale Parameter)
     LDA #A
     STA B
     MEND
```

Beispiel eines MACRO-Aufrufes:

SAMPLE 145,LOC (145 und LOC sind aktuelle Parameter)

Dadurch wird erzeugt:

```
LDA #145
STA LOC
```

Wie Sie sehen können, sind die formalen Parameter durch die aktuellen ersetzt worden.

Die OUT-Anweisung

Um Ihr Listing auf einen Drucker ausgeben zu können, müssen Sie die OUT-Anweisung des Assemblers benützen. Der Text, der direkt nach der OUT-Anweisung kommt, wird ausgedruckt; wenn OUT ohne L,M oder N eingegeben wird, wird die Druckfunktion ausgeschaltet. Das bedeutet, daß kein Text, der diesem Befehl folgt, ausgegeben wird. Wie Sie sehen, gibt Ihnen diese Eigenschaft die Möglichkeit, nur bestimmte Teile des Textes zu drucken.

Die OUT-Anweisung kann drei bis vier Optionen beinhalten. Sie benötigt eine Gerätenummer und manchmal eine Sekundär-Adresse sowie einen File-Namen. "L" bedeutet Listing. D.h. der Ausdruck von OP-Code und dem erzeugten Object-Code. "N" steht für Name-Listing; d.h. alle definierten Marken und MACROs werden ausgedruckt. Wenn eine Marke oder ein MACRO unbenützt sind, druckt der Assembler "UNUSED" hinter die Marke. "M" bedeutet, daß der Macro-Body nicht bei jedem Macro-Aufruf im Listing gedruckt werden soll, sondern nur der einzelne Aufruf mit dem erzeugten Code.

Es gibt noch eine andere Option, die Sie direkt hinter die anderen Optionen setzen können. Diese Option ist "C" ("C" bedeutet Commodore). Sie können dadurch über einen VIC1525 oder einen kompatiblen Drucker ausdrucken.

Ausgabe des Listings auf den Bildschirm:

Wenn Sie das Listing auf dem Bildschirm haben wollen, so müssen Sie folgende Zeile vor dem Text einfügen, den Sie gedruckt haben wollen:

```
      +----- TAB um ein gut lesbares Listing zu erhalten
      :
      :    +---- OUT-Anweisung (Pseudo-OP-Code)
      :    :
      :    :  +--- Assemblerlisting (Möglich)
      :    :  :
      :    :  :+----Listing der Marken (Möglich)
      :    :  ::
      :    :  ::+---- Ausdruck ohne Macros (Möglich)
      :    :  :::
      :    :  :::+--- Komma
      :    :  ::::
      :    :  ::::+----- Gerätenummer (Bildschirm=3)
      :    :  :::::
      :    :  :::::   +---- Zeilenabschluß durch Return
      :    :  :::::   :
<CTRL>-<I>OUT LNM,3<RETURN>
```

Ausdruck mit einem Drucker über den internen (IEEE) Bus:
Um über einen Drucker (meist Gerät #4) drucken zu können, sind folgende Eingaben zu tätigen:

```
      +----- TAB um ein gut lesbares Listing zu erhalten
      :
      :    +---- OUT-Anweisung (Pseudo-OP-Code)
      :    :
      :    :  +--- Assemblerlisting (Möglich)
      :    :  :
      :    :  :+----Listing der Marken (Möglich)
      :    :  ::
      :    :  ::+---- Ausdruck ohne Macros (Möglich)
      :    :  :::
      :    :  :::+--- Komma
      :    :  ::::
      :    :  ::::+----- Gerätenummer (Drucker=4)
      :    :  :::::
      :    :  :::::+--- Komma
      :    :  ::::::
      :    :  ::::::+---- Sekundär-Adresse
      :    :  :::::::     (Siehe Drucker-Manual)
      :    :  :::::::
      :    :  :::::::   +---- Zeilenabschluß durch
      :    :  :::::::   :     Return
      :    :  :::::::   :
<CTRL>-<I>OUT LNM,4,0<RETURN>
```

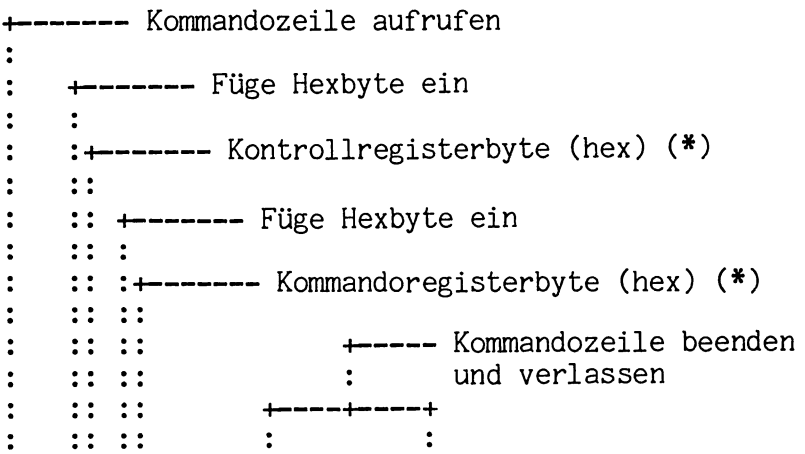
Wenn Sie einen VIC-1525 oder einen kompatiblen Drucker besitzen, geben Sie bitte "C" nach den Optionen (L,N, M) ein.

Ausdruck über das eingebaute RS232-Interface:

Geben Sie dazu folgende Zeile ein:

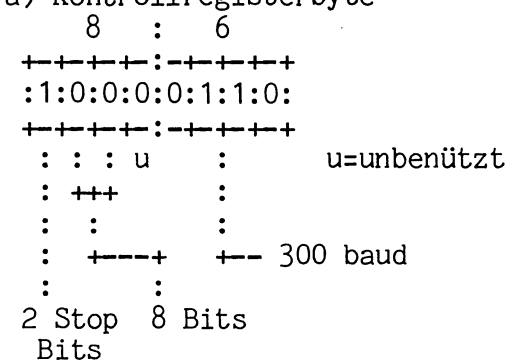

```
<CTRL>-<I>OUT LNM,2,0,"<contrb><comndb>"<RETURN>
```

(*) Geben Sie diese Zeichen bitte folgendermaßen ein:



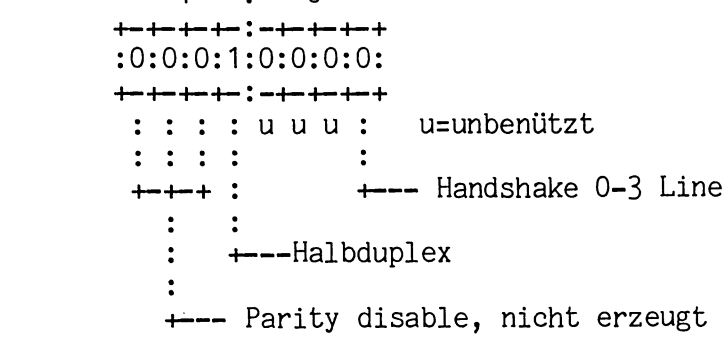
<CTRL>-<A>H86H10<CTRL>-<A><CTRL>-<A>

(*) Durch obiges Beispiel arbeitet das RS232 Interface mit folgender Einstellung:



Siehe auch Commodore Programmer's Reference Guide auf Seite 350

b) Kommandoregisterbyte



Wenn Sie die Spezifikationen der RS232 Schnittstelle ändern wollen, ziehen Sie bitte Ihr Handbuch zu Rate. Das Programm arbeitet normalerweise auf jedem RS232-Drucker. Wir haben es an einem Qume SPRINT 9 Drucker und einem Decwriter getestet. Den Anschluß Ihres Druckers an das RS232-Interface entnehmen Sie bitte dem Manual.
Es ist auch möglich, das Listing auf Diskette oder Cassette zu speichern. Sie brauchen nur die entsprechende Gerätenummer, Sekundäradresse und den Filenamen anzugeben.

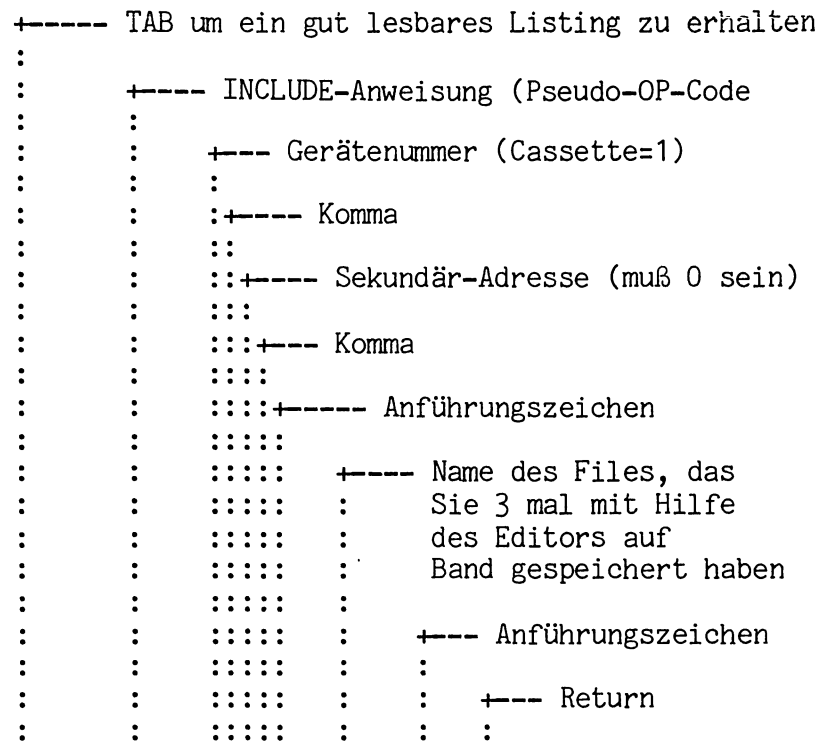
Die INCLUDE-Anweisung

Eine besondere Eigenschaft von MACROFIRE ist die INCLUDE-Anweisung. Sie können dadurch zuvor auf Diskette oder Cassette abgespeicherte Sourcefiles während der Assemblierung am vorhandenen Quelltext des Textpuffers anhängen lassen. Nach Ausführung oder Abbruch der Assemblierung finden Sie den ursprünglichen Textpuffer unverändert vor. Sie können also größere Quelltexte, die die Kapazität des Textpuffers überschreiten würden, assemblieren.

Das Einfügen eines Files von Cassette:

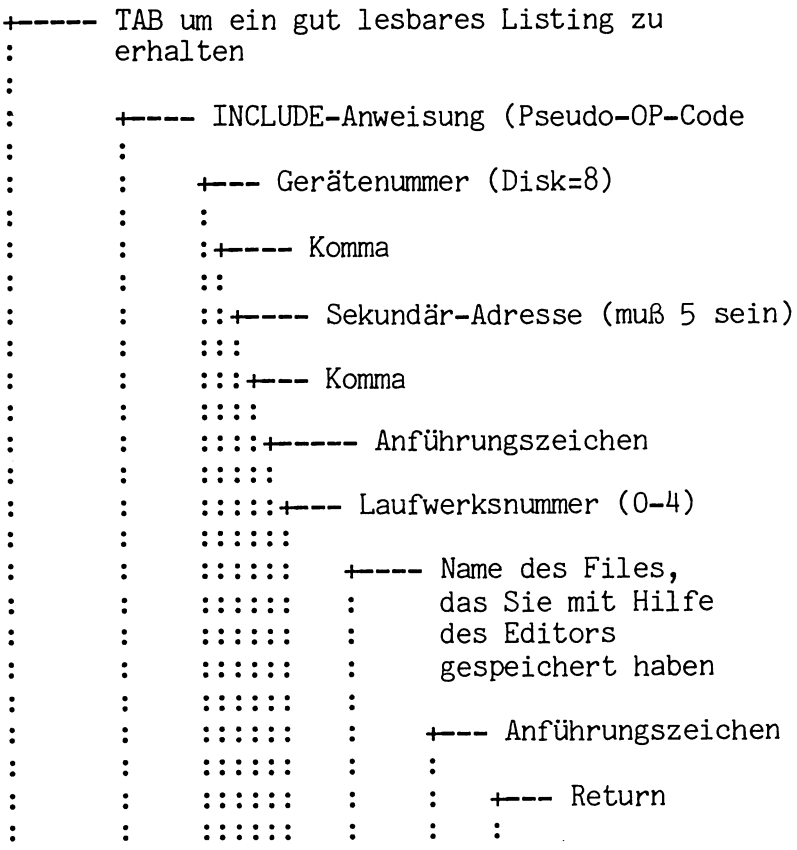
Als erstes müssen Sie den Text, den Sie einfügen wollen, drei mal auf Band abspeichern. Der Assembler hat drei Durchgänge und bei jedem Durchgang wird das File wieder geöffnet.

Das Einfügen der INCLUDE-Anweisung in den Text:
Folgende Zeile muß dafür eingegeben werden:



<CTRL>-<I>INCLUDE 1,0,"FILENAME"<RETURN>

Um die INCLUDE-Anweisung mit der Diskette zu verwenden, ist es nicht nötig, das File 3 mal abzuspeichern! In diesem Fall wird das gleiche File 3 mal entsprechend den 3 Durchgängen geöffnet. Die nächste Abbildung zeigt Ihnen, wie die INCLUDE-Anweisung zusammen mit einer Diskette angewendet wird:



<CTRL>-<I>INCLUDE 8,5,"0:FILENAME"<RETURN>

Es ist möglich, gleichzeitig die INCLUDE- und OUT-Anweisung zu benützen. Während der INCLUDE-Anweisung arbeitet der Assembler verständlicherweise sehr langsam, weil I/O-Operationen sehr langsam sind. Es ist nicht möglich, die INCLUDE-Anweisung zu verschachteln, d.h. Sie können in keinem einzufügenden File eine weitere INCLUDE-Anweisung aufrufen.
Der Gebrauch des Tabulators:
Damit der Text beim Assemblieren und Editieren gut leserlich ist, sollten unbedingt <CTRL>-<I>'s (Tabs) vor dem OP-Code und zwischen dem OP-Code und den Marken eingefügt werden. Alle OP-Codes beginnen dann

in der selben Spalte. Damit Sie aber auch konsequent in der selben Spalte stehen, ist es ratsam, Marken mit maximal acht Zeichen zu verwenden.
z.B.:
Tippen Sie folgendes Programm ein:

```
<CTRL>-<I>OUT LN,3<RETURN>
BSOUT<CTRL><I>EQU $FFD2<RETURN>
<RETURN>
<CTRL>-<I>ORG $B000<RETURN>
<RETURN>
START<CTRL>-<I>LDX #32<RETURN>
LOOP<CTRL>-<I>TXA<RETURN>
<CTRL>-<I>JSR BSOUT<RETURN>
<CTRL>-<I>INX<RETURN>
<CTRL>-<I>BPL LOOP<RETURN>
<CTRL>-<I>BRK<RETURN>
```

Wenn Sie nun den Assembler mit <CTRL>-<Y> starten, erscheint das Listing mit aufgelisteten Marken auf dem Bildschirm; drücken Sie danach irgendeine Taste, worauf die Copyright-Meldung erscheint. Durch einen weiteren Tastendruck kommen Sie wieder an den Anfang des Textes. Geben Sie nun <CTRL>-<P> ein, um den Monitor aufzurufen. Das Programm beginnt bei Adresse $B000 (Hexadezimal). Geben Sie GB000 ein, um das Programm auszuführen. Eine Anzahl von Buchstaben erscheint nun auf dem Bildschirm. Durch die Eingabe von X gelangen Sie wieder über die Copyright-Meldung zum Editor.
Wohin soll man Programme während der Assemblierung dumpen?
Für den Benützer ist ein fester Bereich reserviert. Er reicht von Adresse $B000 bis $CFFF (Hexadezimal). Aber A C H T U N G !! Dieser Bereich wird von I/O-Operationen (Disk, Cassette, RS232) benützt. Der Speicher von Adresse $0800 bis $AFFF wird vom Assembler und Editor benützt. Die Zero-Page-Adressen $50 bis $8F dürfen nicht zerstört werden, wenn man wieder in den Editor zurück will.
DIE MACROs UND IHRE ANWENDUNG
Die Möglichkeit, MACROs zu definieren stellt, ebenso wie Unterprogramme, eine einfache Möglichkeit dar, oft gebrauchte Routinen auf einfachste Weise aufzurufen.
Obwohl MACROs und Unterprogramme sehr ähnlich sind, gibt es dennoch einige Unterschiede, die Sie kennen sollten, um den bestmöglichen Einsatz zu gewährleisten. Sowohl Unterprogramme, als auch MACROs werden vom Programmierer nur

ein einziges Mal geschrieben. Der Assembler fügt dagegen den ganzen MACRO jedesmal dann ins Programm ein, wenn er aufgerufen wird.
Im allgemeinen können wir sagen: MACROs benötigen mehr Speicher, aber sie sind schneller. Unterprogramme sind dagegen leichter in der Anwendung.
Am Anfang des Hauptprogrammes, wo der MACRO definiert wird, verwenden wir formale Parameter. Diese formalen Parameter werden an Stelle der aktuellen Parameter, die jedesmal beim Aufrufen übergeben werden, eingesetzt. Parameter, die nur innerhalb des MACROs benützt werden, heißen lokale Parameter.
z.B.:
Definition eines MACROs:

```
INC2    MACRO PT
        INC PT
        BNE G@
        INC PT+1
G@      EQU *
        MEND
```

Der Name des MACROs ist INC2.
Der Name des formalen Parameters ist PT.
Eine lokale Marke ist G@.
@ bewirkt, daß bei jedem MACRO-Aufruf eine spezifische Marke gesetzt wird.

MACRO-Aufruf:

```
        INC2 AUX
```

Der Name des aktuellen Parameters ist AUX.
Ein anderes Beispiel zeigt eine Blocktransfer-Routine. Die Routine verschiebt einen Speicherblock zwischen STARTP und ENDEP an die Stelle, die mit INTOP beginnt.

```
MOV     LDA STARTP
        CMP ENDP
        BNE CONT
        LDA STARTP+1
        CMP ENDP+1
        BEQ ENDMOV
CONT    LDX #0
        LDA (STARTP,X)
        STA (INTOP,X)

        INC STARTP      -+
        BNE INCINTOP     +-- INC2 STARTP
```

```
                INC STARTP+1  -+

INCINTOPINC INTOP     -+
        BNE JUMP       +-- INC2 INTOP
        INC INTOP+1   -+

JUMP    JMP MOV
ENDMOV  BRK
```

Wenn Sie unser Beispiel-MACRO am Anfang definiert haben, so können Sie die Teile des Programms, die durch Klammern markiert sind, ersetzen, indem Sie die MACROs INC2 STARTP und INC2 INTOP aufrufen.

Fehlermeldungen:
Wenn ein Fehler während der Assemblierung auftritt, wird eine Fehlermeldung ausgegeben, und der Assembler stopt. Drücken Sie bitte eine Taste, um zur Copyright-Meldung und eine weitere, um wieder in den Editor zu kommen. Der Cursor steht jetzt direkt hinter dem fehlerhaften Textteil, so daß Sie sofort mit dem Verbessern anfangen können. Anders ist es, wenn Sie mit der INCLUDE-Anweisung arbeiten, und dabei ein Fehler auftritt. In diesem Fall wird der Cursor hinter die INCLUDE-Anweisung gesetzt, mit der das entsprechende fehlerhafte File aufgerufen worden ist. Die meisten Fehlermeldungen erklären sich selbst. Wir wollen dennoch einige Ausnahmen behandeln.

LINE TOO LONG
Die Zeile besteht aus mehr als 127 Zeichen.

OPCODE DIFFERENT
Zwischen dem zweiten und dritten Durchgang ist ein OPCODE als verschieden erkannt worden. Z.B.: eine Marke ist zuerst als absolute Marke erkannt worden; im dritten Durchgang stellte sich aber heraus, daß sie eine Zero-Page-Anweisung ist. In diesem Fall müssen Sie die Marken vorsichtiger definieren. Eine andere Möglichkeit ist, daß zwei ORG-Commandos in Konflikt geraten.

WARNINGS DURING ASSEMBLY
Beim Listen von assembliertem Code kann eine Warnung auftreten:

"WARNING OPERAND OVERFLOW"
Dies geschieht, wenn Sie einen Ausdruck aus zwei Bytes in einer Speicherstelle, die nur aus einem Byte besteht, ablegen wollen. Dies muß aber nicht unbedingt ein Fehler sein, da es unter Umständen erwünscht ist. Wenn keine Warnungen auftreten, gibt der Assembler "NO WARNINGS" aus.

Beschreibung des eingebauten Monitors in MACROFIRE

Vom Editor aus koennen Sie ueber den Befehl <CTRL>-<P> in den eingebauten Monitor "Supermon" von Jim Butterfield springen. Die Rueckkehr zum Editor erfolgt durch einfaches Druecken der Taste X.

1. SUPERMON 64 ist ein leistungsfähiger Monitor in Maschinensprache

Supermonitor für den C-64

Supermon 64 ist ein Monitor-Programm für den C-64. Er eignet sich besonders für die Entwicklung von Programmen in Maschinensprache und ermöglicht die Ankopplung dieser Programme an BASIC.

Um die Leistungsfähigkeit dieses Monitorprogrames noch zu erhöhen, ist es mit einem Disassembler ausgerüstet.

Befehle, die Sie über das C-64 Keyboard eingeben, starten im Monitor sofort die Programmausführung.

1. Anzeigen oder Ändern von Speicherplätzen oder Registern
2. Breakpoints setzen
3. Laden und Abspeichern von Operationscodeprogrammen (LOAD und SAVE) auf Cassette und Diskette
4. Disassemblieren

Bei der Änderung von Speicherzelleninhalten führt der Monitor eine automatische Lese-/Nachschreibe-Operation durch, damit gesichert ist, daß man in eine vorhandene Speicherzelle geschrieben hat.

Der SUPERMON enthält auch einige Unterprogramme, welche durch das Anwenderprogramm aufgerufen werden können.

Diese Routinen sind:

1. Lesen und Schreiben von Zeichen auf dem Bildschirm
2. Hexadezimaleingabe eines Bytes und Eingabe einer CRLF-Reihenfolge

M = Display Memory (Speicherinhaltsanzeigen)
R = Display Register (Registerinhaltsanzeigen)
G = GO (Beginn mit der Programmausführung)
X = Übergang zu BASIC
S = SAVE
D = xxxx. Disassemble, beginnend an Adresse xxxx
L = LOAD (Laden eines Programmes)
(ein Arbeiten mit dem Drucker vom Monitor aus ist nicht ohne weiteres moeglich !)

Disassemblieren beim SUPERMON geschieht durch Eingabe von D 0000 Return. Jetzt wird von Adresse 0000 hex an eine Bildschirmfüllung voll disassembliert. Erneutes Eingeben von D bringt den nächsten Bildschirminhalt. Return wiederholt die Disassemblierung dieser Seite.

Beispiele:

```
: M C000 C010
: C000 1D C7 48 C6 35 CC FF C7
: C008 C5 CA DF CA 70 CF 23 CB
: C010 9C C8 9C C7 74 C7 IF C8
```

Im Befehl zur Anzeige eines Speicherinhaltes müssen Anfangsadresse und Endadresse genau angegeben werden (4 Digit Hexadezimalzahlen).

Um einen Speicherinhalt zu ändern, fahren Sie mit dem Cursor auf dem Bildschirm an die gew. Zahl, ändern diese und drücken die RETURN- Taste.

```
.R
:  PC SR AC XR YR SP
 C6ED 00 80 00 20 F5
```

Bei jedem Eintritt in den SUPERMON oder beim Verlassen des Monitors werden die Register zwischengespeichert und aufbewahrt. Sie können genau so, wie die Speicherplätze im obigen Beispiel mit dem Cursor geändert werden.

.G C38B

Der GO-Befehl zum Starten eines Programmes kann wahlweise mit oder ohne Anfangsadresse verwendet werden. Achtung ! Bitte beachten Sie die Speicherbelegungstabelle am Ende des Buches, damit Sie nicht in einen Bereich hineinassemblieren, der vom Assembler oder vom Monitor oder vom Quelltext etc. bereits belegt wird.

X

bringt Sie wieder in den Editor von Macrofire.

.L"NAME" lädt von Cassette

Beim LOAD-Befehl darf nichts weggelassen werden! Der File-Name muß unbedingt angegeben werden.

Das Betriebssystem antwortet wie in BASIC mit "PRESS PLAY ON TAPE#1".

.L"NAME",08 lädt von Diskette

Die Speicheradresse wird so geladen, wie Sie in der File-Adresse angegeben wurde. Sie mußte vorher natürlich im SAVE-Befehl eingegeben worden sein.

Unterprogramme in Maschinensprache können auch vom BASIC her geladen werden. Es muß jedoch darauf geachtet werden, daß keine BASIC-Variablen verwendet werden, da der entsprechende Pointer dann hinter das zuletzt geladene Byte gesetzt würde.

.S"PROGRAMM NAME",01,0800,0C80

PRESS PLAY + RECORD ON TAPE#1

Alle Daten, inkl. Anfangs- und Endadresse müssen angegeben werden. Um einen Befehl wieder rückgängig zu machen, geben Sie einfache RETURN ein oder drücken die STOP-Taste.

.S"PROGRAMM NAME",08, 0800,0C80

speichert auf Disk.

Interrupt und Breakpoint-Betrieb

BRK ist ein Software-Unterbrechungs-Befehl, welcher die CPU veranlaßt, die Befehlsausführung sofort zu stoppen. Programmzähler und P-Register werden im STACK (Stapelspeicher) zwischengespeichert (gesichert). Nun springt das Programm durch einen Vektor (Speicherstelle $021B und $021C) an die Adresse, die in diesen beiden Speicherzellen abgelegt ist (HB,LB).

Wenn der Anwender diesen Vektor nicht ändert, bleibt das Programm im Monitor, auch wenn ein BRK-Befehl gegeben wurde. Er druckt dann B* aus und zeigt damit an, daß die Unterbrechung durch einen Breakpoint gekommen ist und nicht durch einen CALL (CALL = C*...).

Auch werden die Register wie nach dem R-Befehl angezeigt. Der Monitor wartet jetzt auf weitere Befehle. Beachten Sie, daß nach einem BRK-Befehl, dessen Vektor

auf den SUPERMON gerichtet ist, der Programmzähler des Anwenders jetzt auf die Speicherzelle zeigt, die unmittelbar nach dem BRK-Befehl kommt.

In jedem Fall sollte der Anwender des BRK-Befehls folgendes beachten:

BRK arbeitet als ein Zwei-Byte-Befehl, wobei der Programmzähler zwei Byte hinter dem BRK-Befehl stehen bleibt. (RTI = Return from Interrupt).

Der IRQ (Interrupt Request) ist im C-64 normalerweise auf ein JSR gerichtet, welches die Uhr steuert und das Tastenfeld jede 1/60 Sekunde abfragt. Wenn der Vektor geändert wird, und die Maschinensprachenunterroutine diesen Vektor nicht zwischenspeichert, muß ein Power-on RESET gegeben werden.

$04 - $22 Zero Page
$400- $76C absolute RAM

$23 - $5A sind "Zero Page Locations" im BASIC Eingangs-Puffer. Sie können verwendet werden, wenn BASIC diesen Speicherbereich nicht benützt. Andere Speicherplätze kann man nur mit großer Vorsicht benutzen, da man nie weiß, wo die C-64 Software noch etwas ablegt.

Testen des Monitors

Schalten Sie Ihren C-64 ein und bringen Sie ihn in den gewohnten Befehlsmode.

```
B*  PC SR AC XR 4R SP
  77FE 71 48 E6 00 F6
```

Es können kleine Veränderungen vorkommen, aber im großen und ganzen sieht die Anzeige wie oben aus.

Die Anzeige der Registerinhalte ist, wie schon erwähnt, die erste Anzeige beim Eintreten in den Monitor.

Sie besteht aus B* Registerinhalte: Programmzähler, Prozessorstatus, Akkumulator, X-Index, Y-Index und Stack-Pointer.

Beachten Sie, daß alle Eingaben und Ausgaben in Hexadezimal erfolgen.

Nachdem der Monitor diese Registeranzeige vorgenommen hat, ist er in der Lage, Befehle entgegenzunehmen.

Die CPU-Register können auch mit dem R-Befehl angesehen werden. Der Monitor sollte dann wie oben antworten, jedoch ohne das Asterisk-Zeichen hinter dem B.

Angezeigte Werte können mit dem Cursor geändert werden. Beachten Sie, daß Zwischenräume zwischen Daten und Adressen sein müssen. (4 Digit für Hex-Adressen, 2 Digit für den Byte-Inhalt).

Speicherinhalte können mit dem M-Befehl angezeigt und dann geändert werden.

Wenn Sie . M 0100 0107 eingeben, sehen Sie z. B. folgendes:

: 0100 10 31 30 32 30 00 30 30

Nun können Sie wieder mit dem Cursor ändern und anschließend neu anzeigen.

Starten Sie das Programm mit G C000.

Benutzen Sie den M-Befehl, um das folgende Testprogramm einzugeben. Es druckt den ASCII-Zeichen-Vorrat auf dem Bildschirm. Dann können Sie wieder mit dem Cursor entsprechende Änderungen durchführen.

Das CHSET-Programm wurde so assembliert, daß es den Bereich ab C000 Hex verwendet.

.G C000 Eingabe führt nun das Programm aus.

Der gesamte ASCII-Zeichenvorrat sollte nun auf dem Bildschirm erscheinen.

Achten Sie darauf, daß die Adresse jetzt im Programmzähler steht und das Programm jetzt durch GO ohne nachfolgende Adresse gestartet werden kann.

Nun wollen wir versuchen, dieses kleine Programm mit BASIC zu verbinden. Diese Technik ist sehr interessant und erlaubt das Einbauen von Maschinenprogrammen in die BASIC-Programme.

Zuerst löschen wir den BRK-Befehl in Speicherzelle C015 durch ein RTS (Return Subroutine $60).

Wechseln Sie den USR-Funktions-Vektor in Zelle 785 und 786 so, daß er auf das Unterprogramm, beginnend an Speicherzelle C000 Hex gerichtet ist, indem Sie M 0311 031F eingeben und die entspechend ändern.

Verlassen Sie nun den Monitor durch Eingabe X.

Prüfen Sie nun nach, ob die Ankopplung des Programmes auch erfolgt ist, und geben Sie folgendes ein:

Das Maschinenprogramm kann auch wie folgt gestartet werden:

SYS(49152)

Neben den bisher beschriebenen Funktionen finden wir beim SUPERMON 64 noch einen einfachen Zeilenassembler. Er wird z.B. mit

.A C000 LDA#$00

gestartet.

Er erlaubt die Eingabe von Mnemonics und rechnet selbst die Adressen und Sprünge aus.

Hinweis für Druckerbetrieb:

Ist ein Drucker an der seriellen Schnittstelle angeschlossen, geht man zuerst mit X in BASIC. Jetzt öffnet man mit OPEN 4,4 und CMD 4 das Druckerfile und geht mit SYS 2048 zurück in den Monitor.

Probleme gab es, als ich über einen IEC-Adapter einen Commodore 2022 Drucker anschließen wollte. Nach Rückkehr in den Monitor werden noch die Register ausgedruckt, aber dann "hängt" sich das System auf.

Weitere Funktionen sind:

.H = Hunt Memory

Zum Aufsuchen von Hex- oder ASCII-Zeichen im Speicherbereich.

.H C000 D000'READ

Sucht den ASCII-String READ im Speicherbereich C000 bis D000 hex. Strings bis zu 32 Zeichen Länge können gesucht werden.

.H C000 D000 20 FF D2

Sucht die drei aufeinanderfolgenden Hexbytes 20 FF D2 im Speicher.

Ein Block-Transfer-Befehl kann mit T gestartet werden.

.T C000 C0FF C1FF

Dieser Befehl bringt den Hexcode aus dem Bereich zwischen C000 und C0FF an die neue Anfangsadresse C1FF. Dies ist ein Blocktransfer-Programm und kein Relocator.

Der SUPERMON 64 von Jim Butterfield legt sich nach dem Start mit RUN und BASIC ganz an das Ende des vorhandenen Speicherbereiches. SUPERMON 64 kann überall im Speicher des C-64 arbeiten. Wer den Monitor jedoch an eine andere Stelle haben will, muß nach dem Laden des Monitors und vor RUN das Ende von BASIC verändern.

Dies geschieht durch Verändern des entsprechenden Zeigers in den Zero-Page-Adressen 55 und 56.

Wenn nur Pagegrenzen in Frage kommen, genügt es, die Zelle 56 entsprechend zu setzen. POKE 56, 32 würde den SUPERMON weit unten im Speicher ansiedeln.

Ändern der Farbe Bildschirmausdruck SUPERMON:

Wem der schwarze Ausdruck des SUPERMON nicht gefällt, kann die Farbe leicht in weiß oder in einer anderen gewünschten Farbe wählen.

Hunt nach A9 90 im Speicher und alle Adressen werden angegeben.

Alle 90 in 05 abändern.

094D
09E4
0A32
0A9B
0B2E
0B72
0C43
0D85
0DCF
0FIO
1061

MACROSPRITE

Um Ihnen das Arbeiten mit MACROFIRE und Sprites auf Ihrem Commodore noch einfacher zu machen, wollen wir Ihnen im nachfolgenden Abschnitt eines unserer Geheimnisse verraten.

Das Entwickeln von schnellen und aufregenden Spielprogrammen setzt einen leistungsfähigen Macroassembler und einen guten Spriteeditor voraus.

MACROFIRE haben Sie sicher bereits schon.

Was Ihnen jetzt noch fehlt, ist ein Spriteeditor, mit dem Sie auf einfache Weise auf dem Bildschirm Ihre Sprites erstellen (möglichst mit dem Joystick zeichnen) können.

Noch praktischer für Sie wäre es dann, wenn Sie die Daten für den Sprite dann auf Diskette ablegen und in den MACROFIRE hereinholen könnten. Dort sollten Sie dann gleich an der Stelle im Quelltext stehen, wo der MACROFIRE diese benötigt.

Das nachfolgende BASIC-Programm tut genau dieses! Alles, was Sie also machen müssen, ist es richtig eingeben und den gewünschten Sprite entwerfen. Da dieses BASIC-Programm viele Commodore-Grafikzeichen und Leerzeichen zwischen Anführungszeichen enthält, sollten Sie prüfen, ob Sie nicht die DM 79, 00 für die Diskette mit den Programmbeispielen aus diesem Buch bestellen sollten.

Das Macrosprite Programm ist auf dieser Diskette und Sie sparen sich die Mühe der Eingabe. Wenn Sie jedoch fit sind und die Grafikzeichen beherrschen, dann gleich an die Tastatur und eingetippt.

Die Bedienung des Sprite-Editors

Mit diesem Programm steht Ihnen ein komfortables Werkzeug zur Erstellung von Sprites zur Verfügung.

Hier eine kurze Übersicht über die Möglichkeiten:

1. Normal und Multicolorsprite
2. Definition des Sprites mit Joystick
3. Definition in Farbe
4. Wechsel der Spritefarben und der Farbe des Hintergrundes zu jedem Zeitpunkt über Maschinenprogramm
5. Ein- und Ausblenden eines Gitters in das Definitionsfeld
6. Das Entstehen des echten Sprites kann über Maschinenprogramm simultan zur Definition mitverfolgt werden.
7. Vergrößerung und Verkleinerung des Sprites zu jedem Zeitpunkt möglich
8. Laden und Abspeichern von Sprites auf Diskette und Cassette
9. Ausgabe der Definitionswerte zu jedem Definitionszeitpunkt
10. Ausdrucken des Sprites mit Werten auf Drucker
11. Beim Druck wird Gitter und Multicolormodus berücksichtigt
12. Spiegeln des Sprites in alle Richtungen
13. Pixelweise Verschiebung der Sprites in alle Richtungen

1. Das Programm stellt sich auf den gewählten Modus selbst ein. Im MC-Modus wird in 4 Farben und doppelt breit, sonst in 2 Farben und normal definiert

2. Joystickdefinition. Dadurch größtmögliche Bewegungsfreiheit. Bewegen des Cursors (Kreuz) mit Joystick in alle Richtungen. Durch Drücken des Feuerknopfes wird die Farbe gewechselt (nur wenn Sie mit dem Cursor stehen bleiben). Cursor ist immer sichtbar.

3. Der Definitionssprite und der echte Sprite werden sofort in den gewählten Farben dargestellt.

4. Über die Funktionstasten F1 - F7 können die Farben des Sprites zu jedem Zeitpunkt geändert werden (Voreinstellung: Hintergrund = grau, SC = schwarz, MC#0 = schwarz, MC#1 = schwarz). Bei Wechsel der

Hintergrundfarbe werden nur das Definitionsfeld und der Hintergrund des echten Sprites eingefärbt (nicht der ganze Bildschirm). Dadurch bleibt das Menue immer lesbar. Einfärbungen erfolgen schlagartig (Maschinenprogramm).

5. In das Definitionsfeld kann zur besseren Orientierung ein Gitter eingeblendet werden ('G' = Gitter ein; 'SHIFT' 'G' = Gitter aus).

6. Sie können das Entstehen des Sprites Punkt für Punkt mitverfolgen (Maschinenprogramm).

7. Vergrößerung und Verkleinerung der Sprites in X-, Y-Richtung (rückgängig machen mit 'SHIFT').

8. Der Sprite kann auf Cassette oder Diskette gespeichert werden. Wird ein Sprite geladen, schaltet das Programm automatisch auf den jeweiligen Modus um.

9. Die errechneten Werte des Sprites werden rechts vom Definitions- feld ausgegeben. Das Menue wird dadurch überschrieben. Zurück-, holen des Menues mit 'SHIFT' 'D'.

10. , 11. Der definierte Sprite wird mit Werten ausgedruckt. Ist das Gitter eingeschaltet, wird es mit ausgedruckt. Im Multicolormodus wird der Sprite entsprechend seinen Farben in 3 verschiedenen Grafikzeichen gedruckt. Die Zuordnung Graphiksymbol zu Farb- register wird mit ausgegeben.

12.Der Sprite kann zu jedem Zeitpunkt über die X-Achse, Y-Achse oder punktsymmetrisch gespiegelt werden.

13. Mit den Cursorsteuertasten können Sie den echten Sprite in alle 4 Richtungen punktweise verschieben. Damit können Sie falsche Plazierungen korrigieren, ohne den gesamten Sprite zu löschen (Maschinenprogramm).

Der jeweilige Status, in dem sich das Menue gerade befindet, wird im Menue durch Farbänderung angezeigt.

MACROSPRITE

```
1 REM*****************************ELCOMP SPRITEEDIT
OR*****************************
2 REM*********************************BY R. HEIGENMOSE
R*****************************
3 POKE56,147:POKE52,147:VI=53248:DIMA(64):DIMFA$
(15):R2=250
7 POKE53281,15:PRINT"■":PRINT"■■■■■■■■■S P R I
T E E D I T O R
8 PRINT"■■■■■■■■■■■■■■BY R. HEIGENMOSER"
9 CO(0)=0:CO(1)=1:CO(2)=2:CO(3)=3:FA$(0)="HINTER
GRUND":FA$(1)="FARBE 1"
10 FA$(2)="FARBE 2":FA$(3)="FARBE 3":MC=2:TA=3:P
RINT:PRINT:GOSUB561
11 GOSUB15000:PRINT:PRINT"■TASTE DRUECKEN!■
                "
12 GETA$:IFA$=""THEN12
13 POKEVI+21,0:PRINT"■":INPUT"MULTICOLORMODUS (J
/N) ";MC$
15 IFMC$<>"J"THEN25
20 MC=2:CM=1:POKE38409,1:POKEVI+28,4:R2=160:GOTO
70
25 MC=1:CM=2:POKE38409,0:POKEVI+29,0
70 FORI=39977TO40777STEP40:FORJ=0TO23
75 POKEI+J,0:NEXTJ,I
80 TA=27:PRINT"■":POKE53280,14:POKE53281,15
90 DI=54272:VX=32:VY=58:PO=1065:HY=38912:FA=2:GI
=1
100 CO(0)=12:CO(1)=0:CO(2)=0:CO(3)=0:ZE(0)=250:Z
E(1)=160:ZE(2)=160:ZE(3)=160
491 FORI=1813TO2013STEP40:FORJ=0TO6:POKEI+J+DI,1
5:POKEI+J,160:NEXTJ,I
493 FORI=832TO894:POKEI,0:NEXT
494 POKE39425,250:SYS39430:POKEVI+21,5
496 POKE39425,12:POKE39426,0:POKE39427,0:POKE394
28,0:POKE39429,12:SYS39441
```

```
500 POKE2040,11:POKE2042,13:POKEVI+39,0:POKEVI,3
2:POKEVI+1,58
505 POKEVI+23,0:POKEVI+29,0
507 POKEVI+37,0:POKEVI+38,0:POKEVI+41,0
510 POKEVI+4,4:POKEVI+5,204:POKEVI+16,4
550 POKE1032,103:POKE1032+DI,11:POKE1040,103:POK
E1040+DI,11
552 POKE1344,111:POKE1344+DI,0:POKE1664,111:POKE
1664+DI,0
554 POKE1912,103:POKE1912+DI,11:POKE1920,103:POK
E1920+DI,11
556 POKE1369,111:POKE1369+DI,0:POKE1689,111:POKE
1689+DI,0:GOSUB560:GOTO600
560 PRINT"■"
561 PRINTTAB(TA)"■F1■="FA$(CO(0)):IFMC=2THEN56
3
562 PRINTTAB(TA)"                ":GOTO564
563 PRINTTAB(TA)"■F3■="FA$(CO(1))
564 PRINTTAB(TA)"■F5■="FA$(CO(2))
565 IFMC=2THEN568
566 PRINTTAB(TA)"                ":GOTO570
568 PRINTTAB(TA)"■F7■="FA$(CO(3))
570 PRINTTAB(TA)"■G■ITTER"
572 PRINTTAB(TA)"■N■EUER SCHIRM"
574 PRINTTAB(TA)"■B■ERECHNUNG"
576 PRINTTAB(TA)"■X■-GROESSER"
578 PRINTTAB(TA)"■Y■-GROESSER"
580 PRINTTAB(TA)"■D■ATEN"
582 PRINTTAB(TA)"■L■OAD SPRITE"
584 PRINTTAB(TA)"■S■AVE SPRITE"
586 PRINTTAB(TA)"■P■RINT SPRITE"
587 PRINTTAB(TA)"■*■=SPIEGELN"
588 PRINTTAB(TA)"MC=■E■IN/■A■US":IFMC=2THENP
OKE55926,5:GOTO590
589 POKE55930,5
590 RETURN
600 GETWA$:IFWA$<>""THENGOSUB2000:PRINT"■"
605 JO=NOTPEEK(56320)AND31
610 Y1=-(JOAND1):Y2=(JOAND2)/2:X1=-(JOAND4)/4:X2
=(JOAND8)/8:FE=(JOAND16)/16
612 X1=8*(X1+X2):Y1=8*(Y1+Y2)
614 IFABS(X1)+ABS(Y1)=0THEN655
616 F1=0:F2=0
```

```
620 VX=VX+MC*X1:VY=VY+Y1
630 IFVX<224ANDVX>24THEN635
632 VX=VX-MC*X1
635 IFVY<226ANDVY>50THEN640
637 VY=VY-Y1
640 POKEVI,VX:POKEVI+1,VY
650 PO=1065+(VX-32)/8+40*((VY-58)/8)
655 IF(PEEK(PO+DI)AND15)=0THENPOKEVI+39,1:GOTO66
0
656 POKEVI+39,0
660 IFFE<>1THEN670
665 FORI=0TOMC-1:POKEPO+I,ZE(FA):POKEPO+DI+I,CO(
FA):POKEPO+HY+I,FA:NEXT
667 ZE=(VX-32)/8:BI=ZE-INT(ZE/8)*8:BY=(VY-58)*.3
75+INT(ZE/8)
668 POKE39568,FA:POKE39569,BY:POKE39570,BI:SYS39
580
670 IFJO<>16THEN680
675 F1=1:IFF2=1THENFA=FA+CM:F1=0:F2=0:IFFA=4THEN
FA=0
680 IFJO=0THENIFF1=1THENF2=1
700 GOTO600
2000 SYS37700:IFPEEK(38410)=0THEN2005
2002 SYS39441:IFGI=1THENGOSUB2130
2003 RETURN
2005 PRINT"■":IFWA$<>"■"THEN2030
2010 CO(0)=CO(0)+1:IFCO(0)=16THENCO(0)=0
2020 POKE39425,CO(0):POKE39429,CO(0):SYS39441:PR
INT"■":PRINTTAB(30)FA$(CO(0))
2025 RETURN
2030 IFWA$<>"■"THEN2060
2035 IF MC=1THENRETURN
2040 CO(1)=CO(1)+1:IFCO(1)=16THENCO(1)=0
2050 POKE39426,CO(1):SYS39441:POKEVI+37,CO(1)
2055 PRINT"■■":PRINTTAB(30)FA$(CO(1)):RETURN
2060 IFWA$<>"■"THEN2090
2070 CO(2)=CO(2)+1:IFCO(2)=16THENCO(2)=0
2080 POKE39427,CO(2):SYS39441:POKEVI+41,CO(2)
2085 PRINT"■■■":PRINTTAB(30)FA$(CO(2)):RETURN
2090 IFWA$<>"■"THEN2120
2095 IFMC=1THENRETURN
2100 CO(3)=CO(3)+1:IFCO(3)=16THENCO(3)=0
2110 POKE39428,CO(3):SYS39441:POKEVI+38,CO(3)
```

```
2115 PRINT"":PRINTTAB(30)FA$(CO(3)):RETURN
2120 IFWA$<>"G"THEN2140
2130 POKE39425,250:FORI=39426TO39428:POKEI,160:N
EXT:SYS39430
2132 POKE55523,5:ZE(0)=250:GI=1
2135 FORI=0TO3:POKE39425+I,CO(I):NEXT:RETURN
2140 IFWA$<>"!"THEN2160
2150 FORI=39425TO39428:POKEI,160:NEXT:SYS39430
2152 POKE55523,2:ZE(0)=160:GI=0
2155 FORI=0TO3:POKE39425+I,CO(I):NEXT:RETURN
2160 IFWA$<>"N"THEN2210
2170 PRINT"NEUER BILSCHIRM ?"
2180 GETA$:IFA$=""THEN2180
2190 IFA$="J"THENPRINT"    OK                  ":GOT
O13
2195 PRINT"           |            |     ":RETURN
2210 IFWA$="X"THENPOKEVI+29,4:POKE55643,5:RETURN

2220 IFWA$="*"THENPOKEVI+29,0:POKE55643,2:RETURN

2230 IFWA$="Y"THENPOKEVI+23,4:POKE55683,5:RETURN

2240 IFWA$=" !"THENPOKEVI+23,0:POKE55683,2:RETURN

2250 IFWA$="D"THENGOSUB3500:RETURN
2255 IFWA$<>"¯"THEN2260
2256 GOSUB3800:GOSUB560:POKE55523,L1:POKE55643,L
2:POKE55683,L3
2257 POKEVI+21,5:FORI=1813TO2013STEP40:FORJ=0TO6
:POKEI+J+DI,15:POKEI+J,160
2258 NEXTJ,I:SYS39516:RETURN
2260 IFWA$<>"L"THEN2350
2265 PRINT"SPRITE LADEN ?
    "
2270 GETA$:IFA$=""THEN2270
2271 IFA$<>"J"THEN2310
2272 POKE55763,5:POKEVI+21,1:INPUT"NAME DES SP
RITES: ";N$
2273 PRINT"
"
2274 INPUT"KASSETTE/DISKETTE (K/D) ";SP$
2275 PRINT"":IFSP$="K"THENOPE
N1,1,0,N$:GOTO2288
```

```
2276 IFSP$="D"THENOPEN15,8,15:OPEN1,8,2,N$+",S,R
":GOTO2288
2278 GOTO2273
2288 INPUT#15,AA,BB$,CC,DD:IFAA<20THEN2290
2289 PRINT"";BB$;"                    ":CLOSE1:CLOS
E15:GOTO2320
2290 FORY=0TO3:FI$(Y)=""
2291 GET#1,AA$:IFAA$=CHR$(13)THEN2293
2292 FI$(Y)=FI$(Y)+AA$:GOTO2291
2293 NEXT:GOSUB4500
2294 IFSP$="D"THEN2300
2295 PRINT"       |       |       ":PRINT"
                    "
2300 CLOSE1:CLOSE15:GOSUB5000
2310 PRINT"          |      |              "

2320 POKEVI+21,5:POKE55763,2:RETURN
2350 IFWA$<>"S"THEN2500
2360 PRINT"SPRITE SPEICHERN ?"
     "
2370 GETA$:IFA$=""THEN2370
2380 IFA$<>"J"THEN2470
2385 POKE55803,5:INPUT"NAME DES SPRITES: ";N$
2387 PRINT"
"
2390 INPUT"KASSETTE/DISKETTE (K/D) ";SP$
2395 PRINT"":IFSP$="K"THENOP
EN1,1,1,N$:GOTO2440
2400 IFSP$="D"THENN$="@0:"+N$:OPEN15,8,15:OPEN1,
8,2,N$+",S,W":GOTO2420
2410 GOTO2390
2420 INPUT#15,AA,BB$,CC,DD:IFAA<20THEN2440
2430 PRINT"";BB$;"                    ":CLOSE1:
CLOSE15:GOTO2480
2440 GOSUB4000:FORX=0TO3:PRINT#1,FI$(X):NEXT
2450 CLOSE1:CLOSE15:IFSP$="D"THEN2470
2460 PRINT"       |       |       ":PRINT"
                    "
2470 PRINT"          |      |              "
2480 POKE55803,2:RETURN
2500 IFWA$<>"P"THEN2600
2505 POKE55843,5:X=0:FORI=1TO63:A(I)=PEEK(831+I)
:NEXT
```

```
2507 IFGI=1THENG1=186:GOTO2510
2508 G1=32
2510 OPEN4,4
2512 KO$(1)=CHR$(145)+" ____________________________
 "+CHR$(8)
2513 KO$(2)=CHR$(15)+CHR$(145)+" ____________________
__________"+CHR$(17)+CHR$(15)
2514 PRINT#4,KO$(1)
2515 FORI=39977TO40777STEP40::PR$=CHR$(15)+CHR$(
145)+" "+CHR$(15):ZA$=""
2517 FORJ=0TO23
2520 IFPEEK(I+J)=0THENPR$=PR$+CHR$(146)+CHR$(G1)
:GOTO2535
2530 IFPEEK(I+J)=2THENPR$=PR$+CHR$(18)+CHR$(32):
GOTO2535
2531 IFPEEK(I+J)=1THENPR$=PR$+CHR$(166):GOTO2535

2532 IFPEEK(I+J)=3THENPR$=PR$+CHR$(191)
2535 NEXTJ
2537 ZA$=ZA$+CHR$(146)
2540 FORK=1TO3:ZA$=ZA$+CHR$(16)+RIGHT$(STR$(K*4+
     40),2)+STR$(A(X+K)):NEXTK:X=X+3
2550 PR$=PR$+CHR$(146)+CHR$(145)+"| "+CHR$(15)+ZA
$+CHR$(8):PRINT#4,PR$:NEXTI
2560 PRINT#4,KO$(2):PRINT#4,CHR$(15)
2570 PRINT#4,CHR$(18)" "CHR$(146)" =SCR"
2572 PRINT#4,CHR$(166)" =SMC #0"
2574 PRINT#4,CHR$(191)" =SMC #1"
2576 CLOSE4
2580 POKE55843,2:RETURN
2600 IFWA$<>"*"THEN2700
2605 POKE55883,5
2610 INPUT"■X-/Y-ACHSE/PUNKTSYMM. (X/Y/P)";SP$
2620 IFSP$="X"THENSYS38500:GOTO2650
2625 IFSP$="P"THENSYS38500:SYS38700:GOTO2650
2630 IFSP$="Y"THENSYS38700:GOTO2650
2640 GOTO2610
2650 SYS39441:IFGI=1THENGOSUB2130
2660 PRINT"■          ■|          :                "
:POKE55883,2:POKE55603,2:RETURN
2700 RETURN
3500 REM DATAS
```

```
3505 L1=PEEK(55523):L2=PEEK(55643):L3=PEEK(55683
)
3510 POKEVI+21,1:GOSUB3800
3515 FORI=1TO63:A(I)=PEEK(831+I):NEXT
3520 PRINT"■":FORI=0TO20:PRINT
3530 FORJ=0TO2:PRINT"■";TAB(26+J*4)A(I*3+J+1):NE
XTJ
3540 NEXTI:POKE55723,2:POKE55603,2:RETURN
3800 REM CLEAR MENUE
3810 PRINT"■":FORI=1TO23:PRINTTAB(26)"
   ":NEXT:RETURN
4000 FORY=0TO3:FI$(Y)="■DFB "
4020 FORX=1+Y*16TO16+Y*16
4025 IFX=64THEN4060
4030 AA$=MID$(STR$(PEEK(831+X)),2)+",":FI$(Y)=FI
$(Y)+AA$
4040 NEXTX:FI$(Y)=LEFT$(FI$(Y),LEN(FI$(Y))-1)
4045 POKE831+X,AA
4050 NEXTY
4060 FI$(Y)=FI$(Y)+MID$(STR$(MC),2):RETURN
4500 POKE55603,5:FORY=0TO3:FI$(Y)=RIGHT$(FI$(Y),
LEN(FI$(Y))-5):AB=1
4520 FORX=1+Y*16TO16+Y*16
4530 AA=VAL(MID$(FI$(Y),AB)):AB=AB+LEN(STR$(AA))

4540 POKE831+X,AA:NEXTX
4550 NEXTY
4560 MC=PEEK(895):RETURN
5000 REM GELADENEN SPRITE DARSTELLEN
5005 POKE38409,MC-1
5007 POKEVI,32:POKEVI+1,58:VX=32:VY=58:PO=1065:F
A=2
5010 IFMC=2THENCM=1:POKEVI+28,4:GOTO5020
5015 CM=2:POKEVI+28,0
5020 SYS38250:CO(1)=1:CO(2)=0:CO(3)=2:SYS39441:I
FGI=1THENGOSUB2130
5600 GOSUB2050:GOSUB2080:GOSUB2110:GOSUB560:IFGI
=0THENPOKE55523,2
5610 IFPEEK(VI+29)=4THENPOKE55643,5
5620 IFPEEK(VI+23)=4THENPOKE55683,5
5630 RETURN
9000 DATASCHWARZ  ,WEISS    ,ROT      ,TUERKIS
,VIOLETT  ,GRUEN    ,"BLAU    "
```

```
9010 DATAGELB      ,ORANGE    ,BRAUN     ,HELLROT
,GRAU 1    ,GRAU 2    ,HELLGRUEN
9020 DATAHELLBLAU ,"GRAU 3    "
10000 DATA195,0,0,231,0,0,126,0,0,60,0,0,60,0,0,
126,0,0,231,0,0,195,0,0
11000 REM MASCH UP GITTER/FARBE
11010 DATA169,4,133,140,169,41,133,139,76,25,154
,169,216,133,140,169
11020 DATA41,133,139,169,156,133,142,169,41,133,
141,162,0,160
11030 DATA0,161,141,170,189,1,154,162,0,129,139,
200,230,139,230
11040 DATA141,208,4,230,140,230,142,192,24,208,2
31,160,0,24,165
11050 DATA139,105,16,133,139,133,141,144,4,230,1
40,230,142,165,142,201,159
11060 DATA208,208,165,141,201,113,208,202,234
11100 DATA162,0,160,0,169,21,133,139,169,219,133
,140,234,234,234
11110 DATA234,234,234,234,234,234,234,234,234,17
3,5,154,129,139,200,230,139
11120 DATA192,7,208,244,160,0,24,165,139,105,33,
133,139,144,233,96
11200 DATA234,234,234,234
11205 DATA0,0,0,0,128,64,32,16,8,4,2,1,24,174,14
5,154,172,146,154
11210 DATA169,0,141,147,154,173,28,208,240,7,238
,147,154
11220 DATA46,144,154,200,238,147,154,110,144,154
,185,148,154
11230 DATA24,110,144,154,176,8,73,255,61,64,3,76
,206,154
11240 DATA29,64,3,157,64,3,206,147,154,240,4,136
,76,186,154,96
11300 REM SPIEGELN X
11310 DATA169,40,133,139,169,156,133,140,169,72,
133,141,169,159,133,142,162
11320 DATA10,160,24,177,139,141,11,150,177,141,1
45,139,173,11,150,145,141,136
11330 DATA208,239,165,139,24,105,40,133,139,144,
2,230,140,165,141,56,233,40
11340 DATA133,141,176,2,198,142,160,24,202,208,2
12,169,63,133,139,169,3,133,140
```

```
11350 DATA133,142,169,123,133,141,160,3,162,10,1
77,139,141,11,150,177,141,145
11360 DATA139,173,11,150,145,141,136,208,239,165
,139,24,105,3,133,139,165
11370 DATA141,56,233,3,133,141,160,3,202,208,220
,96
11400 REM SPIEGELN Y
11410 DATA169,40,133,139,169,156,133,140,133,142
,169,53,133,141,169,21,141
11420 DATA15,150,160,12,162,0,177,139,141,11,150
,161,141,145,139,173,11,150
11430 DATA129,141,136,240,14,24,165,141,105,1,13
3,141,144,2,230,142,76,67,151
11440 DATA160,12,206,15,150,240,29,165,139,24,10
5,40,133,139,144,2,230,140
11450 DATA165,140,133,142,165,139,24,105,13,133,
141,144,2,230,142,76,67,151
11460 DATA169,63,133,139,169,3,133,140,169,21,14
1,15,150,162,8,160,3,177
11470 DATA139,141,16,150,169,0,141,17,150,24,173
,9,150,208,63,173,16,150,106
11480 DATA141,16,150,173,17,150,42,141,17,150,20
2,208,239,162,8,173,17,150
11490 DATA153,10,150,136,208,212,160,3,173,11,
150,145,139,136,173
11500 DATA12,150,145,139,136,173,13,150,145,139,
206,15,150,208,1,96,165,139
11510 DATA24,105,3,133,139,76,147,151,173,16,150
,106,106,141,16,150,173,17
11520 DATA150,42,141,17,150,173,16,150,10,173,17
,150,42,141,17,150,202
11530 DATA202,208,226,76,184,151
11600 REM BEWEGUNG
11610 DATA169,1,141,10,150,165,203,201,7,240,10,
201,2,240,14,169,0,141,10
11620 DATA150,96,173,141,2,240,14,76,216,147,173
,141,2,240,3,76,68,148,76
11630 DATA205,148,169,32,133,139,169,72,133,141,
169,159,133,140,133,142,162,20
11640 DATA160,24,177,139,145,141,136,208,249,165
,140,133,142,165,139,133
11650 DATA141,56,233,40,133,139,176,2,198,140,16
0,24,202,208,227,162,24,169
```

```
11660 DATA0,157,40,156,202,208,250,169,120,133,1
39,169,123,133,141,169,3
11670 DATA133,140,133,142,160,3,162,20,177,139,1
45,141,136,208,249,165,139
11680 DATA133,141,56,233,3,133,139,160,3,202,208
,235,169,0,141,64,3,141,65
11690 DATA3,141,66,3,96,169,80,133,139,169,40,13
3,141,169,156,133,140,133,142
11700 DATA162,20,160,24,177,139,145,141,136,208,
249,165,140,133,142,165,139
11710 DATA133,141,24,105,40,133,139,144,2,230,14
0,160,24,202,208,227,162,24
11720 DATA169,0,157,72,159,202,208,250,169,66,13
3,139,169,63,133,141,169,3,133
11730 DATA140,133,142,160,3,162,20,177,139,145,1
41,136,208,249,165,139,133
11740 DATA141,24,105,3,133,139,160,3,202,208,235
,169,0,141,124,3,141,125,3
11750 DATA141,126,3,96,169,42,133,139,169,41,133
,141,169,156,133,140,133
11760 DATA142,160,0,162,21,169,1,141,15,150,173,
9,150,240,5,169,2,141,15,150
11770 DATA177,139,145,141,200,192,23,208,247,136
,169,0,145,139,160,0,206,15
11780 DATA150,208,235,165,139,24,105,40,133,139,
144,2,230,140,165,141,24,105
11790 DATA40,133,141,144,2,230,142,202,208,195,1
69,63,133,139,169,3,133
11800 DATA140,169,21,141,15,150,162,1,173,9,150,
240,2,162,2,160,3,24,177,139
11810 DATA10,145,139,136,177,139,42,145,139,136,
177,139,42,145,139,202,208,233
11820 DATA165,139,24,105,3,133,139,206,15,150,20
8,212,96,169,40,133,139,169
11830 DATA41,133,141,169,156,133,140,133,142,160
,23,162,21,169,1,141,15,150
11840 DATA173,9,150,240,5,169,2,141,15,150,177,1
39,145,141,136,208,249,200
11850 DATA169,0,145,139,160,23,206,15,150,208,23
7,165,139,24,105,40,133,139
11860 DATA144,2,230,140,165,141,24,105,40,133,14
1,144,2,230,142,202,208,197
11870 DATA169,63,133,139,169,3,133,140,169,21,14
1,15,150,162,1,173,9,150,240
```

```
11880 DATA2,162,2,160,1,24,177,139,74,145,139,20
0,177,139,106,145,139,200
11890 DATA177,139,106,145,139,202,208,233,165,13
9,24,105,3,133,139,206,15
11900 DATA150,208,212,96
12000 REM ERSTELLEN GELADENEN SPRITE
12010 DATA162,63,160,24,169,72,133,139,169,159,1
33,140,169,8,141,11,150,189
12020 DATA63,3,141,10,150,173,9,150,240,28,173,1
0,150,41,3,145,139,136,206
12030 DATA11,150,145,139,136,206,11,150,173,10,1
50,74,74,141,10,150,76,179
12040 DATA149,173,10,150,74,141,10,150,169,0,42,
42,145,139,136,206,11,150
12050 DATA173,11,150,208,201,202,240,19,152,208,
184,160,24,165,139,56,233
12060 DATA40,133,139,176,2,198,140,76,118,149,96

15000 PRINT:PRINT"MASCHINENROUTINEN WERDEN GELES
EN!!":PRINT""
15005 FORI=0TO15:READFA$(I):NEXT
15010 B=0:FORI=704TO727:READA:B=B+A:POKEI,A:NEXT

15015 IFB<>1224THENPRINT"■FEHLER ZEILE 10000":ST
OP
15020 B=0:FORI=39430TO39642:READA:B=B+A:POKEI,A:
NEXT
15025 IFB<>28531THENPRINT"■FEHLER ZEILE 11000-11
240":STOP
15030 B=0:FORI=38500TO38618:READA:B=B+A:POKEI,A:
NEXT
15040 IFB<>15487THENPRINT"■FEHLER ZEILE 11310-11
370":STOP
15050 B=0:FORI=38700TO38918:READA:B=B+A:POKEI,A:
NEXT
15060 IFB<>25446THENPRINT"■FEHLER ZEILE 11410-11
530":STOP
15100 B=0:FORI=37700TO38227:READA:B=B+A:POKEI,A:
NEXT
15110 IFB<>66428THENPRINT"■FEHLER ZEILE 11610-11
900":STOP
15200 B=0:FORI=38250TO38350:READA:B=B+A:POKEI,A:
NEXT
```

```
15220 IFB<>11835THENPRINT"FEHLER ZEILE 12010-12
060":STOP
15500 RETURN
READY.
```

Beispielprogramme in 6502 Maschinensprache

Um Ihnen das Lernen der 6502 Maschinensprache auf Ihrem C-64 so interessant wie moeglich zu gestalten, haben wir im nachfolgenden Teil eine grosse Anzahl sehr interessanter und immer wieder benoetigter Beispiele zusammengestellt. Alle Beispiele sind auf Macrofire in dessen Editor geschrieben worden. Sie koennen diese eingeben und selbst assemblieren und den Objektcode im Monitor mit G starten. Weiterhin koennen Sie diese Programme in Ihre eigenen Programmentwicklungen einbauen.

Spritebehandlung in Maschinensprache

Was den C-64 unter anderem so attraktiv macht, is die Möglichkeit, mit Sprites zu arbeiten. Das Wort Sprite kommt aus dem angelsächsischen Sprachraum und bedeutet soviel wie Elfe. Der Name ist nicht unpassend, denn mit Sprites lassen sich wirklich zauberhafte Wesen erstellen, die das Benutzen auf vielfältige Weise bewegen und verändern kann.

Wenn Sie sich schon einige Zeit mit der Programmierung von Personal Computern beschäftigt haben, werden Sie sicher festgestellt haben, daß Sie mit der Verwendung von Programmiersprachen wie BASIC sehr bald an Grenzen stoßen, was die Geschwindigkeit von bestimmten Abläufen betrifft. Dies gilt ganz besonders fuer die Erstellung von Spielen und hier wieder speziell fuer die Arbeit mit Sprites.

Bei Sprites, die aus normalen Zeichen aufgebaut sind, simulieren Sie einen Bewegungsablauf, indem Sie ein spezielles Zeichen um eine Bildschirmstelle verschieben und das Zeichen an der alten Position löschen.

Das erweckt beim Betrachten den Anschein, als bewege sich dieses Zeichen. Diese Bewegung erfolgt allerdings ziemlich ruckartig, da die Zeichenverschiebung eine Verschiebung um 8 Bit (entspricht 8 sichtbaren Bildpunkten) ist.

Nicht so bei Sprites. Sie können bildpunktweise verschoben werden. Das ergibt einen fließenden Bewegungsablauf; der Zeitaufwand zur Abarbeitung des Vorganges ist dafür aber 4 x so hoch, da Sie zwar die alte Bildposition nicht löschen, dafür aber statt einer Verschiebung 8 Verschiebungen vornehmen müssen.

Sie koennen sich sicher vorstellen, dass ein Spiel in BASIC, das mit mehreren Sprites arbeitet, unerträglich langsam. wenn nicht unmöglich wird. Abhilfe schafft hier nur der Einbau von Maschinenroutinen oder das Programmieen des gesamten Spiels in Maschinensprache.

Mit dem Thema Sprites und Maschinensprache wollen wir uns im folenden befassen.

Zu Anfang ein Überblick über die im C-64 implementierten Möglichkeiten, mit Sprites zu arbeiten.

1. Maximalanzahl der gleichzeitig darstellbaren Sprites = 8 (Reg. 53269)

2. Anzahl der fuer Sprites definierbaren Figurenmuster: theor. mit 255 (Realität bei ca. 50-100, je nach Programmlänge).

3. Jedes Sprite kann einfarbig oder mehrfarbig (3 Farben) definiert werden. (Register 53276).

4. Vergrösserung jedes Sprites in X und/oder Y-Richtung möglich (Reg.53277, 53271).

5. Prioritäten der Sprites untereinander nach aufsteigender Spritenummer, d.h., das Sprite mit der Nummer 0 überdeckt alle anderen Sprites.

6. Priorität Sprite-Hintergrund fuer jedes einzelne Sprite wählbar,d.h., Sprite kann Zeichen überdecken oder umgekehrt (Register 53275).

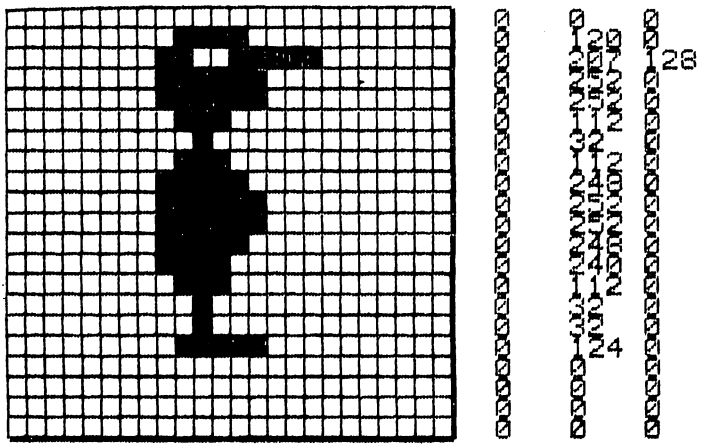
7. Srite-Srite-Kollision. Zusammenstoß zweier oder mehrerer Sprites wird registriert (Register 53278).

8. Srite-Background-Kollision. Zusammenstoß eines oder mehrerer Sprites mit 'Normalzeichen' wird registriert (Register 53279).

Entstehung eines Sprites Schritt für Schritt

Zu Anfang sollten Sie sich im Klaren sein, wie Ihr Sprite aussehen soll. Wir wollen hier keine Elfe, sondern einen Kobold entstehen lassen. Der Kobold sieht gerastert folgendermaßen aus:

Figur 1: Kobold stehend



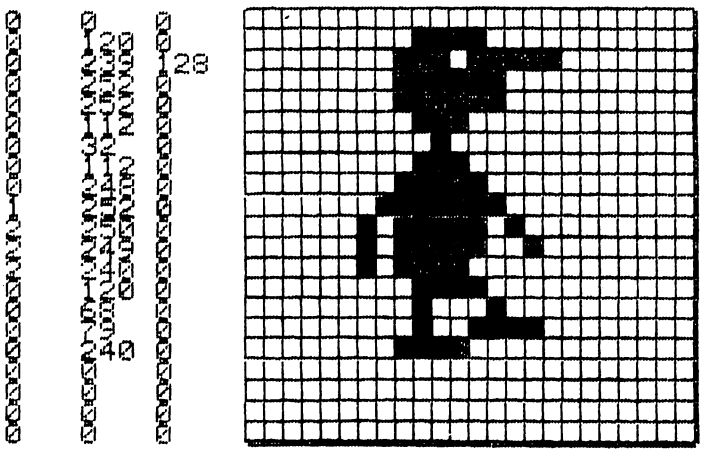
Die entsprechenden Werte (Berechnung siehe Handbuch Seite 70/71) stehen rechts neben dem Raster. Diese Werte werden nun in einem sogenannten Block abgespeichert. Ein Block besteht aus 64 Bytes. Die Anfangsadresse eines Blocks muss durch 64 ohne Rest

teilbar sein. Die so entstandene Zahl ergibt die Blocknummer, mit der das entsprechende Figurenmuster, wenn nötig, aufgerufen wird. Wir haben uns entschlossen, diese Figur in Block 33 einzubringen.

33 * 64 = 2112, d.h., das Muster wird ab Adresse 2112 abgespeichert.
2112 = 0
2113 = 0
2114 = 0
2115 = 0
2116 = 120
2117 = 0 u.s.w.

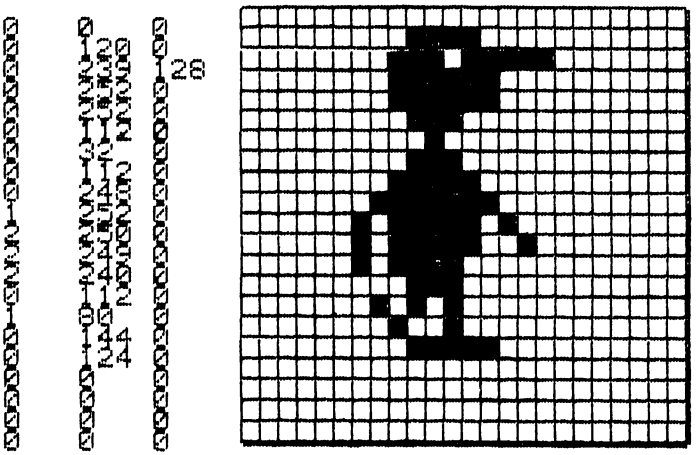
Jetzt merken wir uns Block Nr. 33 entspricht Kobold stehend. Wenn wir dieses Sprite jetzt bewegen, dann sieht es aus, als würde er auf einem Fließband stehen. Wir wollen ihn aber gehen und sogar huepfen lassen. Dazu benötigen wir weitere Figuren:

Figur 2: Kobold Fuß vor



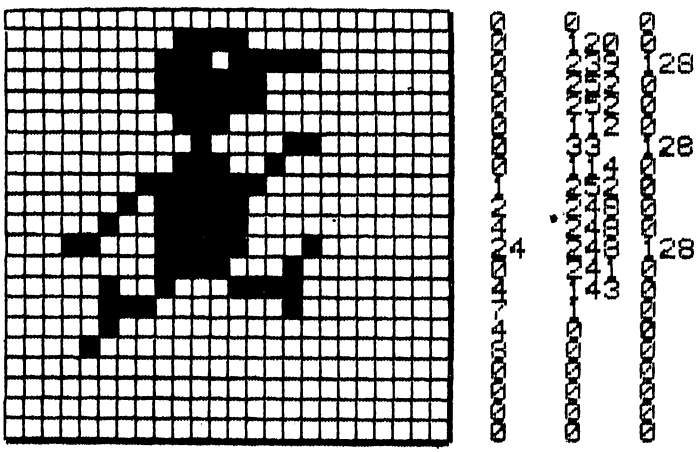
Diese Figur bringen wir in Block 34, also ab Adresse 34 * 64 = 2176 unter.

Figur 3: Kobold Fuß zurück



Figur 3 nach Block 35 entspricht Anfangadresse 2240.

Figur 4: Kobold springt



Figur 4 nach Block 36 entspricht Anfangsadresse 2304

Soll der Kobold auch nach links laufen können, so müssen jetzt alle Figuren jeweils spiegelverkehrt definiert und in entsprechende Blöcke abgespeichert werden (Blocks 38-42).

Für eine Demonstration der wichtigsten Erscheinungen bei der Arbeit mit Sprites habe ich mir folgende Sequenz überlegt.

Der Kobold steht am linken Ende einer Bildschirmzeile. Durch Bewegung des Joysticks nach rechts oder durch Drücken der Taste '>' setzt er sich nach rechts in Bewegung. In der Mitte der Zeile steht ein Block aus Normalzeichen (Space-Reverse). Bewegt sich der Kobold nach rechts, geht er vor dem Block vorbei, bei Linksbewegung geht er hinter dem Block vorbei. (Priorität Sprite-Hintergrund).

Am rechten Zeilenende steht wiederum ein Block aus Normalzeichen. Diesen gilt es zu erreichen. Berührt er ihn (Sprite-Background-Collision), dann plaziert sich der Block ans linke Zeilenende. Behindert wird das Sprite duch ein Faß, daß von rechts heranrollt. Über das Faß kann es durch Drücken der Feuertaste auf den Joystick oder Drücken der Space-Taste auf der Tastatur hinwegspringen. Berührt er das Faß (Sprite-Sprite-Kollision), dann wird er überrollt. Die Sequenz wird mit jeder Berührung des Blocks schneller.

Das folgende Maschinenprogramm soll eine Anregung geben, wie man mit Sprites mittels Maschinensprache umgehen kann.

Es erhebt keinen Anspruch auf perfekte Programmiertechnik, im Gegenteil, es wurde Wert darauf gelegt, die Abläufe strukturiert und übersichtlich darzustellen. Und das geht meist auf Kosten der Eleganz.

Das Programm erklärt sich eigentlich von selber. Trotzdem noch einige Bemerkungen dazu. In den Tabellen ab 832, 840 und 860 sind die Blocknummern für die einzelnen Figuren abgespeichert und zwar paarweise.

Wird eine Bewegung vom Joyport her erkannt, dann wird aus der entsprechenden Tabelle die Blocknummer für die relevante Figur geholt und dem Sprite zugeteilt (fuer Sprite #0 bedeutet das Abspeicherung der Blocknummer nach 2040). Im nächsten Durchgang wird die folgende Blocknummer geholt, die mit der vorhergehenden aber identisch ist. Das Sprite schiebt sich also nur ein Stück vorwärts, ohne sichtbar die Figur zu wechseln.

Die Verzögerungsschleife wird nach jeder Abfrage 45 * 255 = 11475 mal leer durchlaufen. Würde diese Schleife wegfallen, dann könnten Sie das Sprite bei Bewegung nur nochals schwarzen Schatten über den Bildschirm flitzen sehen. Dann sehen Sie, mit welcher Geschwindigkeit Sprite-Programmierung in Maschinensprche möglich wäre.

Jetzt noch einige Bemerkungen zu Sprites allgemein.

Sprite-Sprite-Kollision:

Arbeiten Sie mit mehreren Sprites gleichzeitig, so kann die Abfrage auf Kollision eines Sprites mit einem ganz bestimmten, anderen Sprite u.U. schwierig werden.

Ein Beispiel:
Auf dem Bildschirm stehen 4 Sprites mit Nr. 0 - 3. Sie wollen feststellen, ob Sprite # 0 mit Sprite # 2 kollidiert ist.

Überlappen sich nun diese 2 Sprites, d.h., steht im Register 53278 eine 5, dann haben Sie Glück gehabt und wissen Bescheid.

Sind aber auch Sprite # 1 und Sprite # 3 kollidiert, dann steht im Register eine 15 und Sie können beim besten Willen nicht sagen, welches Sprite mit welchem Kontakt hat.

Beide Kollisionsregister werden übrigens durch Auslegen (PEEK in BASIC und LDA, LDX, LDY in Assembler) auf Null zurückgesetzt. Benötigen Sie den ausgelesenen Wert längere Zeit, dann ist es von Vorteil, ihn zwischenzuspeichern.

Priorität Sprite-Hintergrund

Wenn Sie auf dem Bildschirm Zeichen im Multicolormodus verwenden, dann kann es passieren, daß das Sprite hinter einem Zeichen durchscheint, obwohl die Zeichenfarbe diesen Bildschirmplatz belegt.

Das liegt daran, daß der Sprite nur an den Stellen verdeckt wird, an denen im Bitmuster des Zeichens eine Eins steht. Im Multicolor-Modus werden aber jeweils 2 Bit zu einer Farbe zusammengefaßt.

Verlegen des Video Interface Chip-Adressierungsbereihes

Verlegen Sie den VIC-Adressierungsbereich, dann sollten Sie daran denken, daß mit ihm auch alle Sprite-Blöcke und Blockspeicheradressen verlegt werden. Das wird am deutlichsten an einem Beispiel.

Wenn Sie den Rechner einschalten, dann liegt der Bildschirm ab 1024 bis 2023.

Die Ansprungadressen für den VIC-Baustein, aus denen er sich die relevanten Blocknummern für die entsprechenden Sprites holt, liegen ab 2040 - 2047 für Sprite # 0 - 7. Ein bestimmter Block, in dem Sie das Spritefigurenmuster ablegen, liegt ab (Blocknummer * 64), d.h., Block Nr. 33 liegt z.B. ab Adresse 2112.

Verlegen Sie jetzt den VIC-Adressierungsbereich (und damit den Bildschirm) z.B. nach 16384, dann liegt ihr Bildschirm jetzt von 17408 (= 16384 + 1024) bis 18407 (= 16384 + 2023).

Die Blocknummeradressen für den VIC liegen immer hinter dem momentanen Bildschirm und damit jetzt von 18424 - 18431.

Block Nr. 33 wird jetzt nicht mehr von Adresse 2112 ab abgerufen, sondern von Adresse 16384 + 33 * 64 = 18496.

Verlegen des Adressierungsbereiches des VIC auf Adresse 16384: POKE 56576,6

Verlegen des Bildschirms auf Adresse 17408: POKE 53272, 25 od. 24

Damit ist der Zeichensatz automatisch auf 24576 gelegt.

Bildschirm Zeichensatz

0 0 0 1 1 0 0 X Zeile 53272 zweigeteilt

Bildschirm = 16354 + 1 * 1024

Zeichensatz = 16384 + 8 * 1024

Schlußbemerkungen:

Das Maschinenprogramm kann wie ein normales BASIC-Programm geladen und mit RUN gestartet werden. Es erstreckt sich von Adresse 2048 bis Adresse 4017 dez. Die Erstellung des Demonstrationsprogramms wurde auf dem Editor/Assembler MACROFIRE (H. C. Wagner) der Fa. Ing. W. Hofacker vorgenommen.

Die Sprites wurden kreirt, gespiegelt und ausgedruckt mit dem Programm SPRITEEDITOR (R. Heigenmoser) der Fa. Ing. W. Hofacker.

Listing KOBOLD

```
                        ORG 2048,18048     SYS 3200

0800: 0B080A    DFB 11,8,10,0,158,51,50,48,48,0,0,0
0803: 009E33
0806: 323030
0809: 000000


                        ORG 2112,18111     ANFANG DER BLOECKE FUER FIGUREN


0840: 000000    DFB 0,0,0          KOBOLD STEHEND RECHTS BLOCK33
0843: 007800    DFB 0,120,0,0,239,128,0,252,0,0,252,0,0,112,0,0,32,0,0,112
,0
0846: 00EF80
0849: 00FC00
084C: 00FC00
084F: 007000
0852: 002000
0855: 007000
0858: 00F800    DFB 0,248,0,0,252,0,0,252,0,0,248,0,0,240,0,0,112,0,0,32,0
,0,32,0
085B: 00FC00
085E: 00FC00
0861: 00F800
0864: 00F000
0867: 007000
086A: 002000
086D: 002000
0870: 007C00    DFB 0,124,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0873: 000000
0876: 000000
0879: 000000
087C: 000000
087F: 00

0880: 000000    DFB 0,0,0          KOBOLD FUSS VOR RECHTS BLOCK 34
0883: 007800    DFB 0,120,0,0,239,128,0,252,0,0,252,0,0,112,0,0,32,0,0,112
,0
0886: 00EF80
0889: 00FC00
088C: 00FC00
088F: 007000
0892: 002000
0895: 007000
0898: 00F800    DFB 0,248,0,1,252,0,2,250,0,2,249,0,2,240,0,0,120,0,0,68,0
,0,79,0
089B: 01FC00
089E: 02FA00
08A1: 02F900
08A4: 02F000
08A7: 007800
08AA: 004400
08AD: 004F00
08B0: 00F000    DFB 0,240,0,0,0,0,0,0,0,0,0,0,0,0,0,0
08B3: 000000
```

```
08B6: 000000
08B9: 000000
08BC: 000000
08BF: 00

08C0: 000000  DFB 0,0,0          KOBOLD FUSS ZURUECK RECHTS BLOCK 35
08C3: 007800  DFB 0,120,0,0,239,128,0,252,0,0,252,0,0,112,0,0,32,0,0,112
,0
08C6: 00EF80
08C9: 00FC00
08CC: 00FC00
08CF: 007000
08D2: 002000
08D5: 007000
08D8: 00F800  DFB 0,248,0,1,252,0,2,250,0,2,249,0,2,240,0,0,112,0,1,80,0
,0,144,0
08DB: 01FC00
08DE: 02FA00
08E1: 02F900
08E4: 02F000
08E7: 007000
08EA: 015000
08ED: 009000
08F0: 007C00  DFB 0,124,0,0,0,0,0,0,0,0,0,0,0,0,0,0
08F3: 000000
08F6: 000000
08F9: 000000
08FC: 000000
08FF: 00

0900: 000000  DFB 0,0,0          KOBOLD SPRINGEND RECHTS BLOCK 36
0903: 007800  DFB 0,120,0,0,239,128,0,252,0,0,252,0,0,112,0,0,33,128,0,1
14,0
0906: 00EF80
0909: 00FC00
090C: 00FC00
090F: 007000
0912: 002180
0915: 007200
0918: 01FC00  DFB 1,252,0,2,248,0,4,248,0,24,248,128,0,241,0,4,143,0,7,1
,0,4,0,0
091B: 02F800
091E: 04F800
0921: 18F880
0924: 00F100
0927: 048F00
092A: 070100
092D: 040000
0930: 080000  DFB 8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0933: 000000
0936: 000000
0939: 000000
093C: 000000
093F: 00

0940: 000000  DFB 0,0,0          KOBOLD SITZEND RECHTS BLOCK 37
0943: 007800  DFB 0,120,0,0,239,128,0,252,0,0,252,0,0,112,0,0,32,0,0,112
,0
0946: 00EF80
0949: 00FC00
094C: 00FC00
```

```
094F: 007000
0952: 002000
0955: 007000

0958: 00F800 DFB 0,248,0,1,252,64,1,252,128,2,248,128,6,127,128,0,0,128
,0,0,0,0,0,0
095B: 01FC40
095E: 01FC80
0961: 02F880
0964: 067F80
0967: 000080
096A: 000000
096D: 000000
0970: 000000 DFB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0973: 000000
0976: 000000
0979: 000000
097C: 000000
097F: 00

0980: 000000 DFB 0,0,0          KOBOLD STEHEND LINKS BLOCK 38
0983: 001E00 DFB 0,30,0,1,247,0,0,63,0,0,63,0,0,14,0,0,4,0,0,14,0
0986: 01F700
0989: 003F00
098C: 003F00
098F: 000E00
0992: 000400
0995: 000E00
0998: 001F00 DFB 0,31,0,0,63,0,0,63,0,0,31,0,0,15,0,0,14,0,0,4,0,0,4,0
099B: 003F00
099E: 003F00
09A1: 001F00
09A4: 000F00
09A7: 000E00
09AA: 000400
09AD: 000400
09B0: 003E00 DFB 0,62,0,0,0,0,0,0,0,0,0,0,0,0,0,0
09B3: 000000
09B6: 000000
09B9: 000000
09BC: 000000
09BF: 00

09C0: 000000 DFB 0,0,0          KOBOLD FUSS VOR LINKS BLOCK 39
09C3: 001E00 DFB 0,30,0,1,247,0,0,63,0,0,63,0,0,14,0,0,4,0,0,14,0
09C6: 01F700
09C9: 003F00
09CC: 003F00
09CF: 000E00
09D2: 000400
09D5: 000E00
09D8: 001F00 DFB 0,31,0,0,63,128,0,95,64,0,159,64,0,15,64,0,30,0,0,34,0
,0,242,0
09DB: 003F80
09DE: 005F40
09E1: 009F40
09E4: 000F40
09E7: 001E00
09EA: 002200
09ED: 00F200
09F0: 000F00          DFB 0,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0
09F3: 000000
```

```
09F6: 000000
09F9: 000000
09FC: 000000
09FF: 00

0A00: 000000 DFB 0,0,0          KOBOLD FUSS ZURUECK LINKS BLOCK 40
0A03: 001E00 DFB 0,30,0,1,247,0,0,63,0,0,63,0,0,14,0,0,4,0,0,14,0
0A06: 01F700
0A09: 003F00
0A0C: 003F00
0A0F: 000E00
0A12: 000400
0A15: 000E00
0A18: 001F00 DFB 0,31,0,0,63,128,0,95,64,0,159,64,0,15,64,0,14,0,0,10,1
28,0,9,0
0A1B: 003F80
0A1E: 005F40
0A21: 009F40
0A24: 000F40
0A27: 000E00
0A2A: 000A80
0A2D: 000900
0A30: 003E00 DFB 0,62,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0A33: 000000
0A36: 000000
0A39: 000000
0A3C: 000000
0A3F: 00

0A40: 000000 DFB 0,0,0          KOBOLD SPRINGEND LINKS BLOCK 41
0A43: 001E00 DFB 0,30,0,1,247,0,0,63,0,0,63,0,0,14,0,1,132,0,0,78,0
0A46: 01F700
0A49: 003F00
0A4C: 003F00
0A4F: 000E00
0A52: 018400
0A55: 004E00
0A58: 003F80 DFB 0,63,128,0,31,64,0,31,32,1,31,24,0,143,0,0,241,32,0,12
8,224,0,0,32
0A5B: 001F40
0A5E: 001F20
0A61: 011F18
0A64: 008F00
0A67: 00F120
0A6A: 0080E0
0A6D: 000020
0A70: 000010 DFB 0,0,16,0,0,0,0,0,0,0,0,0,0,0,0,0
0A73: 000000
0A76: 000000
0A79: 000000
0A7C: 000000
0A7F: 00

0A80: 000000 DFB 0,0,0          KOBOLD SITZEND LINKS BLOCK 42
0A83: 001E00 DFB 0,30,0,1,247,0,0,63,0,0,63,0,0,14,0,0,4,0,0,14,0
0A86: 01F700
0A89: 003F00
0A8C: 003F00
0A8F: 000E00
0A92: 000400
0A95: 000E00
```

```
0A98: 001F00   DFB 0,31,0,2,63,128,1,63,128,1,31,64,1,254,96,1,0,0,0,0,0,
0,0,0
0A7B: 023F80
0A9E: 013F80
0AA1: 011F40
0AA4: 01FE60
0AA7: 010000
0AAA: 000000
0AAD: 000000
0AB0: 000000   DFB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0AB3: 000000
0AB6: 000000
0AB9: 000000
0ABC: 000000
0ABF: 00

0AC0: 000000   DFB 0,0,0          FASS FIGUR1 BLOCK 43
0AC3: 000000   DFB 0,0,0,0,0,0,0,0,0,0,120,0,1,254,0,3,255,0,7,255,128
0AC6: 000000
0AC9: 000000
0ACC: 007800
0ACF: 01FE00
0AD2: 03FF00
0AD5: 07FF80
0AD8: 07FF80   DFB 7,255,128,15,255,192,8,0,64,15,255,192,7,255,128,7,255
,128,3,255,0
0ADB: 0FFFC0
0ADE: 080040
0AE1: 0FFFC0
0AE4: 07FF80
0AE7: 07FF80
0AEA: 03FF00
0AED: 01FE00   DFB 1,254,0,0,120,0,0,0,0,0,0,0,0,0,0,0,0,0
0AF0: 007800
0AF3: 000000
0AF6: 000000
0AF9: 000000
0AFC: 000000
0AFF: 00

0B00: 000000   DFB 0,0,0          FASS FIGUR2 BLOCK 44
0B03: 000000   DFB 0,0,0,0,0,0,0,0,0,0,120,0,1,254,0,3,255,0,7,249,128
0B06: 000000
0B09: 000000
0B0C: 007800
0B0F: 01FE00
0B12: 03FF00
0B15: 07F980
0B18: 07F380   DFB 7,243,128,15,231,192,15,207,192,15,159,192,7,63,128,6,
127,128
0B1B: 0FE7C0
0B1E: 0FCFC0
0B21: 0F9FC0
0B24: 073F80
0B27: 067F80
0B2A: 03FF00   DFB 3,255,0,1,254,0,0,120,0,0,0,0,0,0,0,0,0,0,0,0,0
0B2D: 01FE00
0B30: 007800
0B33: 000000
0B36: 000000
0B39: 000000
```

```
0B3C: 000000
0B3F: 00

0B40: 000000  DFB 0,0,0          FASS FIGUR3 BLOCK 45
0B43: 000000  DFB 0,0,0,0,0,0,0,0,0,0,120,0,1,206,0,3,207,0,7,207,128,7,
207,128
0B46: 000000
0B49: 000000
0B4C: 007800
0B4F: 01CE00
0B52: 03CF00
0B55: 07CF80
0B58: 07CF80
0B5B: 0FCFC0  DFB 15,207,192,15,207,192,15,207,192,7,207,128,7,207,128,3
,207,0
0B5E: 0FCFC0
0B61: 0FCFC0
0B64: 07CF80
0B67: 07CF80
0B6A: 03CF00
0B6D: 01CE00  DFB 1,206,0,0,120,0,0,0,0,0,0,0,0,0,0,0,0,0
0B70: 007800
0B73: 000000
0B76: 000000
0B79: 000000
0B7C: 000000
0B7F: 00

0B80: 000000  DFB 0,0,0          FASS FIGUR4 BLOCK 46
0B83: 000000  DFB 0,0,0,0,0,0,0,0,0,0,120,0,1,254,0,3,255,0,6,127,128,7,
63,128
0B86: 000000
0B89: 000000
0B8C: 007800
0B8F: 01FE00
0B92: 03FF00
0B95: 067F80
0B98: 073F80
0B9B: 0F9FC0  DFB 15,159,192,15,207,192,15,231,192,7,243,128,7,249,128,3
,255,0
0B9E: 0FCFC0
0BA1: 0FE7C0
0BA4: 07F380
0BA7: 07F980
0BAA: 03FF00
0BAD: 01FE00  DFB 1,254,0,0,120,0,0,0,0,0,0,0,0,0,0,0,0,0
0BB0: 007800
0BB3: 000000
0BB6: 000000
0BB9: 000000
0BBC: 000000
0BBF: 00


                ORG 7200,19199     ANFANG HAUPTPROGRAMM
                CLEAR    EQU $E544  BILDSCHIRM LOESCHEN
                KOFIRE   EQU 832    832-839 BLOCKNR. DER KOBOLDFIGUREN RECHTS

                KOFILI   EQU 840    840-847 BLOCKNR. DER KOBOLDFIGUREN LINKS
                ZAEHLO   EQU 850    WAEHLT BENOETIGTE FIGUR AUS
                FASSFI   EQU 860    860-871 BLOCKNR. DER FASSFIGUREN
```

```
                ZAEHFA   EQU 875      WAEHLT BENOETIGTE FASSFIGUR AUS
                JOYST    EQU 56320    CONTROL PORT 2 JOYSTICK
                TASTE    EQU 203      TASTENABFRAGE
                XREG     EQU 53248    X-KOORDINATEN SPRITE #0
                YREG     EQU 53249    Y-      -"-      -"-
                HIGBIT   EQU 53264    HOECHSTES BIT
                SPRUFL   EQU 876      ZEIGT AN OB IM SPRUNG(1) ODER NICHT(0)
                FASSFL   EQU 877      ZEIGT AN OB FASS GERADE ROLLT
                YVERZ    EQU 878      VERZOEGERUNG
                SPZAOB   EQU 879      ZAEHLER FUER SPRUNG NACH OBEN
                SPZAUN   EQU 880      ZAEHLER FUER FALL NACH UNTEN
                ZUFALL   EQU 57495    ZUFALLSROUTINE


0C80: A92D              LDA #45
0C82: 8D6E03            STA YVERZ     VERZOEGERUNG VOREINSTELLUNG
0C85: A921              LDA #33       BLOCKNR. DER FIGUREN (33-47) WERDEN IN
0C87: 8D4003            STA 832       ENTSPRECHENDE TABELLE GELADEN
0C8A: 8D4103            STA 833
0C8D: A922              LDA #34,      KOBOLD RECHTS FUSS VOR
0C8F: 8D4203            STA 834
0C92: 8D4303            STA 835
0C95: A923              LDA #35       KOB. RECHTS FUSS ZURUECK
0C97: 8D4403            STA 836
0C9A: 8D4503            STA 837
0C9D: A926              LDA #38       KOB. LINKS STEHEND
0C9F: 8D4803            STA 840
0CA2: 8D4903            STA 841
0CA5: A927              LDA #39       KOB. LINKS FUSS VOR
0CA7: 8D4A03            STA 842
0CAA: 8D4B03            STA 843
0CAD: A928              LDA #40       KOB. LINKS FUSS ZURUECK
0CAF: 8D4C03            STA 844
0CB2: 8D4D03            STA 845
0CB5: A92B              LDA #43       FASSFIGUREN
0CB7: 8D5C03            STA 860
0CBA: 8D5D03            STA 861
0CBD: A92C              LDA #44
0CBF: 8D5E03            STA 862
0CC2: 8D5F03            STA 863
0CC5: A92D              LDA #45
0CC7: 8D6003            STA 864
0CCA: 8D6103            STA 865
0CCD: A92E              LDA #46
0CCF: 8D6203            STA 866
0CD2: 8D6303            STA 867
0CD5: 2044E5            JSR CLEAR     BIDSCHIRM LOESCHEN
0CD8: A90C              LDA #12
0CDA: 8D21D0            STA 53281     BILDSCHIRMFARBE GRAU2
0CDD: A918    INIT      LDA #24
0CDF: 8D00D0            STA XREG      KOORDINATENREGISTER X,Y VON
0CE2: A9D7              LDA #215      SPRITE #0 VORBESETZEN
0CE4: 8D01D0            STA YREG
0CE7: A941              LDA #65
0CE9: 8D02D0            STA 53250     KOORDINATENREG. VON SPRITE #1
0CEC: A9E8              LDA #232      =FASS VORBESETZEN
0CEE: 8D03D0            STA 53251
0CF1: A902              LDA #2
0CF3: 8D10D0            STA HIGBIT    HOECHSTES BIT FUER FASS SETZEN
0CF6: A9A0              LDA #160      SPACE REV. ZEILENMITTE U. ZEILENENDE
0CF8: 8DA807            STA 1960
```

```
0CFB: 8DA907          STA 1961
0CFE: 8DD007          STA 2000
0D01: 8DD107          STA 2001
0D04: 8DBE07          STA 1982
0D07: 8DBF07          STA 1983
0D0A: 8DE607          STA 2022
0D0D: 8DE707          STA 2023
0D10: A902            LDA #2      FARBE ROT FUER SPACE REV. ZEILENMITTE
0D12: 8DA8DB          STA 56232
0D15: 8DA9DB          STA 56233
0D18: 8DD0DB          STA 56272
0D1B: 8DD1DB          STA 56273
0D1E: A907            LDA #7      FARBE GELB ZEILENENDE
0D20: 8DBEDB          STA 56254
0D23: 8DBFDB          STA 56255
0D26: 8DE6DB          STA 56294
0D29: 8DE7DB          STA 56295
0D2C: 8D98DB          STA 56216   GELB ZEILENANFANG
0D2F: 8D99DB          STA 56217
0D32: 8DC0DB          STA 56256
0D35: 8DC1DB          STA 56257
0D38: A921            LDA #33
0D3A: 8DF807          STA 2040    SPRITE #0=KOBOLD MIT FIG. 1 VORBESETZEN
0D3D: A901            LDA #1
0D3F: 8D17D0          STA 53271-- KOBOLD IN Y-RICHTUNG VERGROESSERN
0D42: 8D1DD0          STA 53277   IN X-RICHTUNG VERGROESSERN
0D45: A900            LDA #0
0D47: 8D27D0          STA 53287   FARBE KOBOLD IST SCHWARZ
0D4A: A909            LDA #9
0D4C: 8D28D0          STA 53288   FARBE FASS=BRAUN
0D4F: A901            LDA #1
0D51: 8D15D0          STA 53269   KOBOLD ERSCHEINT
0D54: A900            LDA #0
0D56: 8D6C03          STA SPRUFL  SPRUNGFLAG,ZAEHLER MIT 0 VORBES.
0D59: 8D5203          STA ZAEHKO  ZAEHLER KOB./FASSFIGUREN AUF NULL
0D5C: 8D6B03          STA ZAEHFA
0D5F: 8D6F03          STA SPZA0B  ZAEHLER SPRUNG AUF NULL
0D62: 8D7003          STA SPZAUN
0D65: 8D05D4          STA 54277   ANSCHLAG,ABSCHWELLEN HART
0D68: A9F0            LDA #240
0D6A: 8D06D4          STA 54278   HALTEN LAUT
0D6D: A90F            LDA #15
0D6F: 8D18D4          STA 54296   LAUTSTAERKE VOLL
0D72: AD5203 BEWEGUNG LDA ZAEHKO  UEBERPRUEFUNG OB LETZTE FIGUR ERREICHT
0D75: C906            CMP #6      DANN WIEDER VON VORNE
0D77: D005            BNE ZAEHOK
0D79: A900            LDA #0
0D7B: 8D5203          STA ZAEHKO
0D7E: AD00DC ZAEHOK   LDA JOYST
0D81: 49FF            EOR #255
0D83: 291F            AND #31
0D85: AA              TAX
0D86: 2910            AND #16     FEUERKNOPFABFRAGE
0D88: D006            BNE FIRE    JA!
0D8A: A5CB            LDA TASTE
0D8C: C93C            CMP #60     SPACE TASTE GEDRUECKT?
0D8E: D005            BNE NOFIRE  NEIN!
0D90: A901    FIRE    LDA #1
0D92: 8D6C03          STA SPRUFL  1 ZEIGT SPRUNGBEDINGUNG AN
0D95: AD6C03 NOFIRE   LDA SPRUFL
0D98: F006            BEQ ABFRAG  KEIN SPRUNG
```

```
0D9A: 20440F          JSR SPRUNG    SPRUNGROUTINE
0D9D: 4C1C0E          JMP FASS
0DA0: 8A      ABFRAG  TXA           ABFRAGE JOYSTICK AUF LINKS
0DA1: 2904            AND #4
0DA3: D006            BNE MARK1     JA!
0DA5: A5CB            LDA TASTE
0DA7: C92F            CMP #47       ABFRAGE TASTATUR AUF '<'
0DA9: D033            BNE RECHTS    NEIN!
0DAB: A901    MARK1   LDA #1
0DAD: 8D1BD0          STA 53275     SPRITE HINTER ZEICHEN
0DB0: AD10D0          LDA HIGBIT    IN RECHTER BILDHAELFTE?
0DB3: 2901            AND #1
0DB5: D00D            BNE REBILD
0DB7: AD00D0          LDA XREG
0DBA: 38              SEC           VERSCHIEBEN SPRITE 1 LINKS
0DBB: E901            SBC #1
0DBD: C904            CMP #4        LINKER RAND DANN STOP
0DBF: 901D            BCC RECHTS
0DC1: 4CCF0D          JMP LINEIN
0DC4: AD00D0 REBILD   LDA XREG      SPRITE #0 1 PUNKT LINKS
0DC7: 38              SEC
0DC8: E901            SBC #1
0DCA: B003            BCS LINEIN    SEITENUEBERSCHREITUNG?
0DCC: CE10D0          DEC HIGBIT    HOECHSTES BIT AUF NULL
0DCF: 8D00D0 LINEIN   STA XREG      NEUE POS. ABSPEICHERN
0DD2: AC5203          LDY ZAEHKO
0DD5: B94803          LDA KOFILI,Y FIGUR ENTSPRECH. ZAEHLER AUSWAEHLEN
0DD8: 8DF807          STA 2040      ABSPEICHERN!
0DDB: EE5203          INC ZAEHKO    NAECHSTE FIGUR
0DDE: 8A      RECHTS  TXA
0DDF: 2908            AND #8        ABFRAGE JOYST RECHTS
0DE1: D006            BNE MARK2     JA!
0DE3: A5CB            LDA TASTE
0DE5: C92C            CMP #44       ABFRAGE '>'
0DE7: D033            BNE FASS      NEIN!
0DE9: A900    MARK2   LDA #0
0DEB: 8D1BD0          STA 53275     SPRITE VOR ZEICHEN
0DEE: AD10D0          LDA HIGBIT
0DF1: 2901            AND #1        IM LINKEN BILD?
0DF3: F00D            BEQ LIBILD
0DF5: AD00D0          LDA XREG
0DF8: 18              CLC
0DF9: 6901            ADC #1
0DFB: C964            CMP #100      RECHTER RAND ERREICHT DANN STOP
0DFD: B01D            BCS FASS
0DFF: 4C0D0E          JMP RENEIN
0E02: AD00D0 LIBILD   LDA XREG
0E05: 18              CLC
0E06: 6901            ADC #1
0E08: 9003            BCC RENEIN    SEITENUEBERSCHREITUNG?
0E0A: EE10D0          INC HIGBIT    HOECHSTES BIT AUF 1
0E0D: 8D00D0 RENEIN   STA XREG
0E10: AC5203          LDY ZAEHKO
0E13: B94003          LDA KOFIRE,Y FIGUR SELEKTIEREN
0E16: 8DF807          STA 2040      ABSPEICHERN!
0E19: EE5203          INC ZAEHKO


0E1C: AD6D03 FASS     LDA FASSFL
0E1F: D028            BNE FASS1     FASS ROLLT GERADE
0E21: 2097E0          JSR ZUFALL
```

```
0E24: A58F            LDA 143
0E26: 253F            AND 63
0E28: C901            CMP #1        CHANCE 1:63 DASS FASS ENTSTEHT
0E2A: D003            BNE FASS2
0E2C: 4C9C0E          JMP VERZOE    KEIN FASS
0E2F: A941    FASS2   LDA #65
0E31: 8D02D0          STA 53250     FASS KOORD. VORBES.
0E34: A901            LDA #1
0E36: 8D6D03          STA FASSFL    FASS ROLLT
0E39: A902            LDA #2
0E3B: 0D10D0          ORA HIGBIT    FASS IM RECHTEN TEIL PLAZIEREN
0E3E: 8D10D0          STA HIGBIT
0E41: A902            LDA #2
0E43: 0D15D0          ORA 53269     FASS EINBLENDEN
0E46: 8D15D0          STA 53269
0E49: AD10D0 FASS1    LDA HIGBIT    IN RECHTER BILDHAELFTE?
0E4C: 2902            AND #2
0E4E: D021            BNE REBI      JA!
0E50: AD02D0          LDA 53250
0E53: 38              SEC           VERSCHIEBEN SPRITE#1 UM 2 PUNKTE LINKS
0E54: E902            SBC #2
0E56: C904            CMP #4        LINKER RAND DANN STOP
0E58: B027            BCS FASS3     NEIN!
0E5A: AD10D0          LDA HIGBIT
0E5D: 2902            AND #2        LINKER RAND RECHTE BILDHAELFTE?
0E5F: D020            BNE FASS3     JA!
0E61: A9FD            LDA #253
0E63: 2D15D0          AND 53269     FASS AUSBLENDEN
0E66: 8D15D0          STA 53269
0E69: A900            LDA #0
0E6B: 8D6D03          STA FASSFL    FASS ROLLT NICHT MEHR
0E6E: 4C9C0E          JMP VERZOE
0E71: AD02D0 REBI     LDA 53250     SPRITE #1 2 PUNKTE LINKS
0E74: 38              SEC
0E75: E902            SBC #2
0E77: B008            BCS FASS3     KEINE SEITENUEBERSCHREITUNG
0E79: A9FD            LDA #253
0E7B: 2D10D0          AND HIGBIT
0E7E: 8D10D0          STA HIGBIT    HOECHSTES BIT FASS AUF NULL
0E81: 8D02D0 FASS3    STA 53250     NEUE POS. ABSPEICHERN
0E84: AC6B03          LDY ZAEHFA
0E87: B95C03          LDA FASSFI,Y FIGUR ENTSPRECH. ZAEHLER AUSWAEHLEN
0E8A: 8DF907          STA 2041      ABSPEICHERN!
0E8D: EE6B03          INC ZAEHFA    NAECHSTE FIGUR
0E90: AD6B03          LDA ZAEHFA
0E93: C908            CMP #8        LETZTE FIGUR ERREICHT?
0E95: D005            BNE VERZOE    NEIN!
0E97: A900            LDA #0
0E99: 8D6B03          STA ZAEHFA


0E9C: AC6E03 VERZOE   LDY YVERZ     VERZOEGERUNGSSCHLEIFE
0E9F: A2FF    MAR2    LDX #255      WIRD ANFANGS 11475 MAL DURCHLAUFEN
0EA1: CA      MAR1    DEX
0EA2: D0FD            BNE MAR1
0EA4: 88              DEY
0EA5: D0F8            BNE MAR2


0EA7: AD1ED0 KOLLIS   LDA 53278
0EAA: 2903            AND #3        KOBOLD-FASS-KOLLISION?
```

```
0EAC: C903          CMP #3
0EAE: D003          BNE KOLL1
0EB0: 4C020F        JMP SITZEN
0EB3: AD1FD0 KOLL1  LDA 53279
0EB6: 2901          AND #1          KOBOLD-ZEICHEN-KOLL.?
0EB8: F00E          BEQ KOLL2       NEIN!
0EBA: AD10D0        LDA HIGBIT
0EBD: 2901          AND #1
0EBF: D00A          BNE KOLLRE      MIT RECHTS KOLLIDIERT!
0EC1: AD00D0        LDA XREG
0EC4: C950          CMP #80
0EC6: 9010          BCC KOLLI       MIT LINKS KOLLIDIERT!
0EC8: 4C720D KOLL2  JMP BEWEGUNG    NICHT ODER MIT MITTE KOLLIDIERT
0ECB: A9A0   KOLLRE LDA #160
0ECD: 20E50E        JSR LIN
0ED0: A920          LDA #32
0ED2: 20F50E        JSR REC
0ED5: 4C720D        JMP BEWEGUNG
0ED8: A920   KOLLI  LDA #32
0EDA: 20E50E        JSR LIN
0EDD: A9A0          LDA #160
0EDF: 20F50E        JSR REC
0EE2: 4C720D        JMP BEWEGUNG
0EE5: CE6E03 LIN    DEC YVERZ       LINKS ZEICHEN SETZEN/LOESCHEN
0EE8: 8D9807        STA 1944
0EEB: 8D9907        STA 1945
0EEE: 8DC007        STA 1984
0EF1: 8DC107        STA 1985
0EF4: 60            RTS
0EF5: 8DBE07 REC    STA 1982        RECHTS ZEICHEN SETZEN/LOESCHEN
0EF8: 8DBF07        STA 1983
0EFB: 8DE607        STA 2022
0EFE: 8DE707        STA 2023
0F01: 60            RTS

0F02: A911   SITZEN LDA #17
0F04: 8D04D4        STA 54276       DREIECK STIMME 1
0F07: A90F          LDA #15
0F09: 8D01D4        STA 54273       FREQUENZ=210HZ
0F0C: AD1BD0        LDA 53275
0F0F: 2901          AND #1          FIGUR LINKS/RECHTS?
0F11: F008          BEQ SIT1        RECHTS!
0F13: A92A          LDA #42
0F15: 8DF807        STA 2040
0F18: 4C200F        JMP SIT2
0F1B: A925   SIT1   LDA #37         SITZEND RECHTS
0F1D: 8DF807        STA 2040
0F20: A9DC   SIT2   LDA #220        AUF BODEN SETZEN
0F22: 8D01D0        STA YREG
0F25: 20390F        JSR VER
0F28: 20390F        JSR VER         VERZOEGERUNG
0F2B: 20390F        JSR VER
0F2E: A900          LDA #0
0F30: 8D04D4        STA 54276       SOUND AUS
0F33: 8D6D03        STA FASSFL
0F36: 4CDD0C        JMP INIT        NEU INITIALISIEREN
0F39: A0FF   VER    LDY #255
0F3B: A2FF   SIT4   LDX #255
0F3D: CA     SIT3   DEX
0F3E: D0FD          BNE SIT3
0F40: 88            DEY
0F41: D0F8          BNE SIT4
0F43: 60            RTS
```

```
0F44: A921    SPRUNG   LDA #33          SOUND EIN!
0F46: 8D04D4           STA 54276        WAVE=SAEGEZAHN
0F49: AD1BD0           LDA 53275
0F4C: 2901             AND #1           RICHTUNG LINKS ODER RECHTS
0F4E: F008             BEQ SPRU1
0F50: A929             LDA #41
0F52: 8DF807           STA 2040         KOBOLD SPRUNG LINKS
0F55: 4C5D0F           JMP SPRU2
0F58: A924    SPRU1    LDA #36
0F5A: 8DF807           STA 2040         KOBOLD SPRUNG RECHTS
0F5D: AD6F03  SPRU2    LDA SPZAOB
0F60: C90C             CMP #12          SPRUNGHOEHE ERREICHT?
0F62: F013             BEQ SPRU3        JA!
0F64: AD01D0           LDA YREG
0F67: 38               SEC
0F68: E904             SBC #4           SPRITE #0 4 PUNKTE NACH OBEN
0F6A: 8D01D0           STA YREG
0F6D: 49FF             EOR #255         AKKU KOMPLEMENTIEREN
0F6F: 0A               ASL
0F70: 8D01D4           STA 54273        NACH STIMME1
0F73: EE6F03           INC SPZAOB       ZAEHLER ERHOEHEN
0F76: 60               RTS              ZURUECK
0F77: AD7003  SPRU3    LDA SPZAUN
0F7A: C90C             CMP #12          WIEDER AUF BODEN?
0F7C: F013             BEQ SPRU4        JA!
0F7E: AD01D0           LDA YREG
0F81: 18               CLC
0F82: 6904             ADC #4           SPRITE #0 4 PUNKTE RUNTER
0F84: 8D01D0           STA YREG
0F87: 49FF             EOR #255
0F89: 0A               ASL
0F8A: 8D01D4           STA 54273
0F8D: EE7003           INC SPZAUN
0F90: 60               RTS
0F91: A900    SPRU4    LDA #0           KEIN SPRUNG!
0F93: 8D6C03           STA SPRUFL
0F96: 8D7003           STA SPZAUN
0F99: 8D6F03           STA SPZAOB
0F9C: 8D04D4           STA 54276        SOUND AUS
0F9F: AD1BD0           LDA 53275
0FA2: 2901             AND #1           FIGUR LINKS ODER RECHTS?
0FA4: F006             BEQ SPRU5        RECHTS!
0FA6: A926             LDA #38
0FA8: 8DF807           STA 2040         NORMALE FIGUR LINKS
0FAB: 60               RTS
0FAC: A921    SPRU5    LDA #33
0FAE: 8DF807           STA 2040         NORMALE FIGUR RECHTS
0FB1: 60               RTS
PHYSICAL ENDADDRESS: $4E31

*** NO WARNINGS

CLEAR    $E544    KOLL1S   $0EA7   UNUSED   ZAEHOK   $0D7E
KOFILI   $0348    KOLL2    $0EC8            NOFIRE   $0D95
FASSFI   $035C    KOLLI    $0ED8            MARK1    $0DAB
JOYST    $DC00    REC      $0EF5            LINEIN   $0DCF
YREG     $D000    SIT1     $0F1B            MARK2    $0DE9
HIGBIT   $D010    VER      $0F39            RENEIN   $0E0D
FASSFL   $036D    SIT3     $0F3D            FASS2    $0E2F
SPZAOB   $036F    SPRU1    $0F58            REBI     $0E71
ZUFALL   $E097    SPRU3    $0F77            VERZOE   $0E9C
BEWEGUNG $0D72    SPRU5    $0FAC            MAR1     $0EA1
FIRE     $0D90    KOFIRE   $0340            KOLL1    $0EB3
ABFRAG   $0DA0    ZAEHKO   $0352            KOLLRE   $0EC8
REBILD   $0DC4    ZAEHFA   $0368            LIN      $0EE5
RECHTS   $0DDE    TASTE    $CB              SITZEN   $0F02
LIBILD   $0E02    YREG     $D001            SIT2     $0F20
FASS     $0E1C    SPRUFL   $036C            SIT4     $0F3B
FASS1    $0E49    YVERZ    $036E            SPRUNG   $0F44
FASS3    $0E81    SPZAUN   $0370            SPRU2    $0F5D
MAR2     $0E9F    INIT     $0CDD            SPRU4    $0F91
```

Einfache arithmetische Operationen in Maschinensprache

In diesem Abschnitt sollen einige einfache arithmetische Operationen beschrieben und erklärt werden, die z. B. für die Punkteausgabe bei Spielen und vielerlei anderen Anwendungen außerordentlich wichtig sind.

Den Abschluß bilden drei kleine Maschinenroutinen, die in bereits bestehende Programme eingebaut werden können, und die das Punktehandling und die Ausgabe bei einem Spiel erledigen sollen. Das Kernstück unserer Überlegungen bildet der sogenannte Fließkommaakkumulator (kurz FAC), der ab Adresse 61-66 hex (97 - 102 dez) zur Verfügung steht.

Mit Hilfe dieses Akkus und einiger ROM-Routinen können die wichtigsten arithmetischen Operationen durchgeführt werden.

Eine reelle Zahl wird in diesem Akku in sogenannter halblogarithmischer Darstellung gespeichert, eine Integerzahl belegt 2 Byte der Mantisse und wird in 2-Byte-Adressform dargestellt.

z = m * b^e Zahl = Mantisse * Basis ^ Exponent (bei reeller Zahl)

z = lowbyte + 256 * highbyte (bei Integerzahl)

Der vorzeichenbehaftete Exponent steht in Zelle 61 hex (97 dez), die Mantisse in Zelle 62 - 65 hex (98 - 101 dez) und das Vorzeichen der Zahl in Zelle 66 hex (102 dez). Um welchen der beiden Zahlentypen es sich handelt, wird in Adresse OE hex (14 dez) der Zeropage mitgeteilt.

OE = 80 hex entspricht integer

OE = 00 entspricht reell

Außer dem Fließkommaakku (FAC) steht ein zweiter Akkumulator zur Verfügung, der für arithmetische Operationen benutzt werden kann. Er wird mit 'ARG' bezeichnet und liegt ab Adresse 69 - 6E hex (105 - 110 dez).

Die Belegung von 'ARG' ist analog der von 'FAC'.

Ein dritter Akkumulatortyp kann vom Benutzer selbst definiert werden, d.h., der Anwender kann ihn an eine beliebige Stelle im Speicher setzen. Den dritten Typ wollen wir hier kurz 'VAR' nennen.

Bevor nun die Arbeit mit den drei Akkutypen an einem konkreten Beispiel erklärt wird, zuerst noch die Beschreibung der wichtigsten ROM-Routinen, die die arithmetischen Operationen durchführen.

ROM-Routinen

Adresse hex	dez.	Vorbelegung	Kurzbeschreibung
B3A2		Y	Y -> FAC
Wirkung:	Der Inhalt des Y-Registers wird in Fließkommadarstellung umgewandelt und in 'FAC' abgelegt. (Zahlenbereich 0 - 255).		
B395		A,Y	A,Y -> FAC
Wirkung:	Die Zahl, die sich aus dem Inhalt des A-Registers + 256 * dem Inhalt des Y-Registers ergibt, wird in Fließkomma umgewandelt und in 'FAC' abgelegt. Zahlenbereich 0 - 65535)		
BC9B		-	FAC (reell) -> FAC (int)
Wirkung:	Der Inhalt von 'FAC' wird in 2-Byte-Adressform umgewandelt (Integer-Zahl)		
BBD4		X,Y	FAC -> VAR
Wirkung:	Der Inhalt von 'FAC' wird nach 'VAR' übertragen. In X und Y muß dabei die Anfangsadresse von 'VAR' stehen. Beispiel: Anfangsadresse VAR = 1000 dez: dann: X = 232 , Y = 3		
BBA2		A,Y	VAR -> FAC
Wirkung:	Der Inhalt von 'VAR' wird nach FAC übertragen. In A und Y muß dabei die Anfangsadresse von 'VAR' stehen.		
BA8C		A,Y	VAR -> ARG
Wirkung:	Der Inhalt von 'VAR' wird nach 'ARG' übertragen. In A und Y muß die Anfangsadresse von 'VAR' stehen.		
BBFC		-	FAC : = ARG
Wirkung:	Übertragung des Inhalts von 'ARG' nach 'FAC'.		
BC0C		-	ARG : = FAC
Wirkung:	Übertragung des Inhalts von 'FAC' nach 'ARG'.		
BC5B		A,Y	Vergleich FAC,VAR

Wirkung: Vergleich des Inhalts von 'VAR' und 'FAC'. Das Resultat wird im Akku übergeben. In A,Y muß die Adresse von 'VAR' stehen.

Resultat:
A = 0 bedeutet FAC = VAR
A = 1 bedeutet FAC > VAR
A = 255 dez bedeutet FAC < VAR

BDDD - FAC -> ASCII

Wirkung: Der Inhalt von 'FAC' wird in ASCII-Format umgewandelt und ab Adresse 100 - 10A hex (256 - 266 dez) im Speicher abgelegt. Das Ende des ASCII-Strings wird durch eine Null gekennzeichnet.
Beispiel:
Wert in FAC entspricht 1024
Speicherbelegung ab Adresse 100 hex nach Ansprung der Routine:

Adresse:	256	257	258	259	260	261	...
Inhalt:	32	49	48	50	52	0	
	Kein Vorzeichen, da positiv	1	0	2	4	Ende ASCII-String	

Der Inhalt von FAC wird durch diese Routine nicht verändert.

B86A - FAC: = FAC+ARG

Wirkung: Werte von 'ARG' und 'FAC' werden addiert und in 'FAC' abgelegt.

B867 A,Y FAC: = FAC+VAR

Wirkung: Werte von 'VAR' und 'FAC' werden addiert und in 'FAC' abgelegt. In A und Y muß die Anfangsadresse von 'VAR' stehen.

B853 - FAC: = ARG-FAC

Wirkung: Werte von 'ARG' und 'FAC' werden subtrahiert. Ergebnis steht in 'FAC'.

B850 A,Y FAC = VAR - FAC

Wirkung: Werte von 'VAR' und 'FAC' werden subtrahiert. Ergebnis steht in 'FAC'. In A,Y muß die Anfangsadresse von 'VAR' stehen.

BAE2	-	FAC = FAC * 10

Wirkung: Inhalt von 'FAC' wird mit 10 multipliziert. Ergebnis steht in 'FAC'.

BAFE	-	FAC = FAC/10

Wirkung: Inhalt von 'FAC' wird durch 10 dividiert. Ergebnis steht in 'FAC'.

BB0F	A,Y	FAC: = VAR / FAC

Wirkung: Der Wert von 'VAR' wird durch den Wert von 'FAC' dividiert. Ergebnis steht in 'FAC'. In A und Y muß die Anfangsadresse von 'VAR' stehen.

BB12	-	FAC: = ARG / FAC

Wirkung: Der Wert von 'ARG' wird durch den Wert von 'FAC' geteilt. Ergebnis steht in 'FAC'.

BF78	A,Y	FAC: = VAR ^ FAC

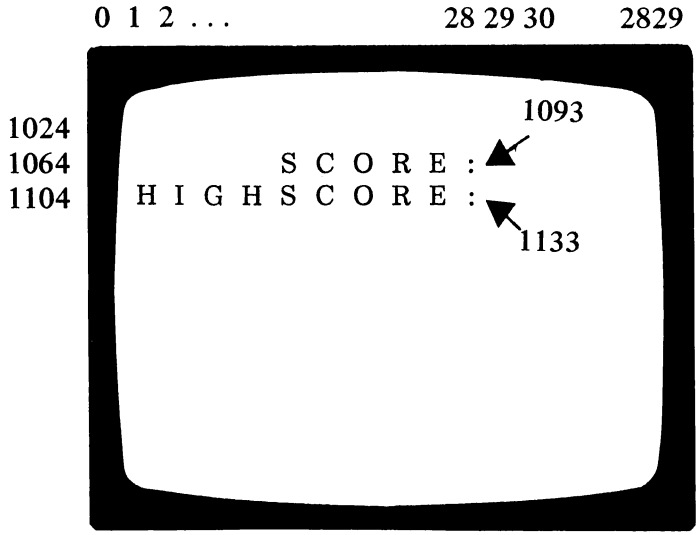
Wirkung: Der Wert von 'VAR' wird mit dem Wert von 'FAC' potenziert. Ergebnis steht in FAC. In A und Y muß die Anfangsadresse von 'VAR' stehen.

BF7B	-	FAC = ARG ^ FAC

Wirkung: Der Wert von 'ARG' wird mit dem Wert von 'FAC' potenziert. Ergebnis steht in 'FAC'.

Die Benutzung dieser ROM-Routinen in Verbindung mit den drei Fließakkumulatortypen soll jetzt am Beispiel der Punktebehandlung in einem Spiel erläutert werden.

Bildschirmausschnitt:



So soll unser Bildkopf aussehen; die Punkte werden ab Bildschirmposition 1093 (entspricht 1024 + 69), die Höchstpunktzahl ab Position 1133 (entspricht 1024 + 109) ausgegeben.

Folgende Forderungen werden an die Punktroutine gestellt:

- Am Anfang soll die Punktzahl mit Null vorbesetzt werden, die Höchstpunktzahl erhält den Wert 1000.

- Die Punktzahl soll bei einem eingetroffenen Ereignis erhöht werden.

- Übertrifft die Punktzahl die Höchstpunktzahl, so soll ein Bonus von 500 Punkten gegeben werden.

- Bei Spielende soll überprüft werden, ob die Punktezahl größer als die Höchstpunktzahl ist.

Falls das zutrifft, soll die Höchstpunktzahl der Punktzahl angeglichen werden.

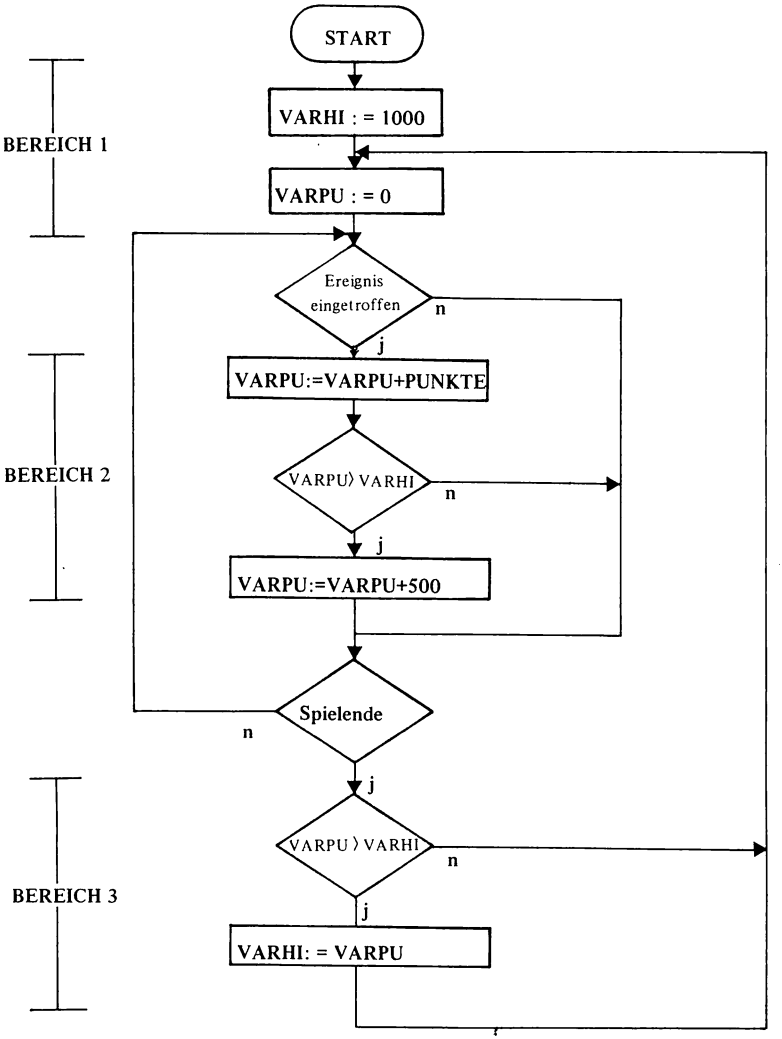
Da die Akkumulatoren 'FAC' und 'ARG' bei Operationen laufend überschrieben werden, definieren wir zuallererst zwei Bereiche, in denen die Punkte und die Höchstpunktzahl stehen sollen.

Z. B. Punktezahl ab Adresse 1000 - 1005 dez
Höchstpunkte ab Adresse 1006 - 1011 dez

Adresse 1000 in 2-Byte-Adressform: 232,3
Adresse 1006 in 2-Byte-Adressform: 238,3

Wir wollen diese zwei selbstdefinierten Fließkommaakkumulatoren vom Typ 'VAR' mit 'VARPU' (für Punkte) und 'VARHI' (für Höchstpunktzahl) benennen.

Schema unseres Spielablaufs:



Da für direktes Laden der Fließkommaakkus vom Typ 'VAR' keine ROM-Routinen zur Verfügung stehen, muß der Umweg über 'FAC' genommen werden.

Bereich 1

```
FLAG    EQU $B000
XREG    EQU $B001
PUNKT   EQU $B002
POS     EQU $B003
FARBE   EQU $B004

        ORG $B010
```

```
**************************************
*                                    *
* BEREICH 1 :  VARPU , VARHI         *
*              INITIALISIEREN        *
*                                    *
**************************************
```

```
B010: A064     LDY #100
B012: 20A2B3   JSR $B3A2    Y NACH FAC
B015: 20E2BA   JSR $BAE2    FAC=FAC*10
B018: A26D     LDX #109     POS. HOECHSTPUNKTZAHL
B01A: 2026B1   JSR FACOUT   AUSGABE
B01D: A2EE     LDX #238     ADRESSE VARHI
B01F: A003     LDY #3
B021: 20D4BB   JSR $BBD4    FAC NACH VARHI
B024: A000     LDY #0
B026: 20A2B3   JSR $B3A2    Y NACH FAC
B029: A245     LDX #69      POS. PUNKTE
B02B: 2026B1   JSR FACOUT   AUSGABE
B02E: A2E8     LDX #232     ADRESSE VARPU
B030: A003     LDY #3
B032: 20D4BB   JSR $BBD4    FAC NACH VARPU
```

Die Hauptunterroutine für die Punktebehandlung besteht aus 3 Teilen.

1. Punkteanzahl wird im Y-Register übergeben und von der Routine aufaddiert.

2. ASCII-String (Punkteanzahl) wird an der Stelle ausgegeben, die im X-Register festgelegt wird.

   In unserem Beispiel wäre dies Bildschirmadresse 1093 dez (entspricht 1024 + 69), d.h. X = 69

3. Überprüfung, ob Highscore übertroffen wurde. Wenn ja, dann Bonus geben und entsprechendes Flag setzen, damit in diesem Spieldurchgang kein zweites Mal Bonus gegeben wird.

Bereich 2

```
                    ORG $B100


                ************************************
                *                                  *
                * BEREICH2 :                       *
                * -  PUNKTEBEHANDLUNG              *
                * -       AUSGABE                  *
                * -  CHECK AUF HIGHSCORE           *
                *                                  *
                ************************************


B100: AC02B0            LDY PUNKT           ROUTINE BEI EINTREFFEN
B103: AE03B0            LDX POS             EINES EREIGNISSES
B106: 2009B1            JSR PUNKTE



B109: 8E01B0 PUNKTE     STX XREG            POSITION ZWISCHENSPEICHERN
B10C: 20A2B3            JSR $B3A2           Y NACH FAC
B10F: A9E8              LDA #232            ADRESSE VARPU
B111: A003              LDY #3
B113: 208CBA            JSR $BA8C           VARPU NACH ARG
B116: 206AB8            JSR $B86A           FAC=ARG+FAC
B119: A2E8              LDX #232            ADRESSE VARPU
B11B: A003              LDY #3
B11D: 20D4BB            JSR $BBD4           VARPU=FAC
B120: 20DDBD            JSR $BDDD           FAC IN ASCII UMWANDELN
```

```
B123: AE01B0          LDX XREG          POSITION LADEN
B126: A000    FACOUT  LDY #0            AUSGABEROUTINE
B128: B90001  LOOP2   LDA $100,Y
B12B: C900            CMP #0            ENDE STRING?
B12D: D003            BNE KON1
B12F: 4C40B1          JMP HICH
B132: 9D0004  KON1    STA 1024,X        AUF BILDPOS. SETZEN
B135: AD04B0          LDA FARBE
B138: 9D00D8          STA 55296,X       FARBE SETZEN
B13B: C8              INY               SCHLEIFENZAEHLER ERHOEHEN
B13C: E8              INX
B13D: 4C28B1          JMP LOOP2
B140: AD00B0  HICH    LDA FLAG          HOECHSTPUNKTE SCHON
B143: D023            BNE HICH1         UEBERTROFFEN?
B145: A9E8            LDA #232          ADRESSE VARPU
B147: A003            LDY #3
B149: 20A2BB          JSR $BBA2         FAC=VARPU
B14C: A9EE            LDA #238          ADRESSE VARHI
B14E: A003            LDY #3
B150: 205BBC          JSR $BC5B         VERGLEICH
B153: 3013            BMI HICH1         FAC (VARPU) <VARHI
B155: A901            LDA #1            FAC >=VARHI
B157: 8D00B0          STA FLAG          FLAG SETZEN
B15A: A0FA            LDY #250
B15C: A245            LDX #69
B15E: 2009B1          JSR PUNKTE        500 PUNKTE AUSGEBEN
B161: A0FA            LDY #250
B163: A245            LDX #69
B165: 2009B1          JSR PUNKTE
B168: 60      HICH1   RTS
```

Routine zur Überprüfung, ob Punktezahl > Höchstpunktezahl

Bereich 3

```
                ORG $B300

*************************************
*                                   *
*  BEREICH 3 :                      *
*  UEBERGABE BEI UEBERTREFFEN       *
*  DER HOECHSTPUNKTZAHL             *
*                                   *
*************************************


B300: A9E8            LDA #232          ADRESSE VARPU
B302: A003            LDY #3
B304: 20A2BB          JSR $BBA2         VARPU NACH FAC
B307: A9EE            LDA #238          ADRESSE VARHI
B309: A003            LDY #3
B30B: 205BBC          JSR $BC5B         VERGLEICH FAC(VARPU) ,VARHI
B30E: 300C            BMI MARK1         FAC < VARHI
B310: A2EE            LDX #238          ADRESSE VARHI
B312: A003            LDY #3
```

```
B314: 20D4BB          JSR $BBD4       FAC NACH VARHI
B317: A26D            LDX #109        POSITION HOECHSTPUNKTZAHL
B319: 2026B1          JSR FACOUT      AUSGABE
B31C: 60      MARK1   RTS
PHYSICAL ENDADDRESS: $B31D

*** NO WARNINGS

FLAG            $B000           XREG            $B001
PUNKT           $B002           POS             $B003
FARBE           $B004           PUNKTE          $B109
FACOUT          $B126           LOOP2           $B128
KON1            $B132           HICH            $B140
HICH1           $B168           MARK1           $B31C
```

NOTIZEN

Beschriften im Graphikmodus

Im Graphikmodus ist es außerordentlich kompliziert, Beschriftungen vorzunehmen. Teilt man den Graphikbildschirm in zwei Hälften, indem man den RASTER-Interrupt verwendet, dann kann man eine Hälfte zwar beschriften, doch ist auf diesem Bildschirmteil keine Graphik mehr möglich.

Einen anderen Weg beschreitet das nachfolgende Maschinenprogramm.

Mit Hilfe dieses Programms können alle im Zeichensatz vorhandenen Zeichen oder aber selbst definierte Zeichen und Graphiksymbole an jeder beliebigen Stelle des Graphikbildschirms ausgegeben werden, d. h. , es gibt 64000 Platzierungsmöglichkeiten. Dadurch ist das Überlappen oder Versetzen von Buchstaben möglich und die Beschriftung kann punktgenau an jede geforderte Stelle gesetzt werden.

Es sei dem Leser überlassen, die Möglichkeiten, die diese Routine birgt, voll auszuschöpfen.

Funktionsweise des Programms: In Zelle $C000 muß die X-Koordinate des Bildpunktes stehen, an dem das Zeichen eingeblendet werden soll. Liegt der Bildpunkt an der rechten Bildschirmhälfte (>255), dann muß Zelle C001 mit 1 (sonst mit 0) vorbesetzt werden.

Analog muß in $C002 die Y-Koordinate des Punktes stehen.

In Zelle C003 und C004 steht die Anfangsadresse des Graphikbildschirms (muß nun einmal eingeschrieben werden).

In Zelle C006 und C007 wird der Routine mitgeteilt, ab welcher Adresse die Bitmuster fuer den Zeichensatz stehen (nur einmal mitteilen). In Zelle (005 schließlich sollte der Bildschirmcode des gewünschten Zeichens stehen.

- Aus der X- und Y-Koordinate wird der gewünschte Punkt im Graphik-Bildschirm errechnet.

- Nun wird die Anfangsadresse der Bitmuster geholt und mit dem Bildschirmcode des gewünschten Zeichens mal 8 (1 Zeichen besteht aus 8 Byte) als Offset die Anfangsadresse des zu kopierenden Zeichens angewählt.

- Die folgenden 8 Byte werden in einen Zwischenspeicher kopiert.

- Nun wird errechnet, um wieviele Pixel der angewählte Bildpunkt von einer normalen Zeichenposition abweicht.

- Das Bitmuster im Zwischenspeicher wird mit Hilfe der errechneten Werte korrigiert und schließlich an der geforderten Stelle in das Graphik-RAM einkopiert.

- Der angewählte Bildpunkt ist der Punkt in der linken, oberen Ecke des dargestellten Zeichens.

Die folgende BASIC-Routine dient zu Test- und Demonstrationszwecken. In ihr wurde der Zeichensatz komplett aus dem ROM nach Adresse 24576 ins RAM kopiert (Zeile 1-6).

Zeile 10 löscht den Graphikbildschirm

Zeile 15 - 17 erwarten Eingaben

Zeile 20 schaltet die hochauflösende Graphik (320 * 200 Punkte) ein und setzt das Graphik-RAM auf 8192.

In Zeile 35 wird der Zeichenroutine der Anfang des GraphiK-RAMS mitgeteilt.

In Zeile 37 erhält die Routine die Nummer des zu kopierenden Zeichens. Außerdem wird ihr der Anfang des Zeichensatzes mitgeteilt.

Zeile 40 erledigt die Vorbesetzungen des Bildpunktes, an dem das Zeichen erstellt werden soll.

In Zeile 60 erfolgt der Ansprung der Zeichenroutine.

Zeile 80 schaltet auf Normalbetrieb um, damit die Eingabe von neuen Werten möglich ist.

```
1 REM ZEICHENSATZ IN RAM (AB 24576) LADEN
2 POKE56333,127:POKE1,51:POKE56,67
3 FORI=53248TO57344
4 POKEI-28672,PEEK(I)
5 NEXT
6 POKE1,55:POKE56333,129
10 FORI=8192TO16192:POKEI,0:NEXT:REM GRAPHIKBILDSCHIRM LOESCHEN
15 A=0:INPUT"X-KOORDINATE";X:IFX>255THENX=X-256:A=1
16 INPUT"Y-KOORDINATE";Y
17 INPUT"ZEICHEN NR.";AS
20 PRINT"□":POKE53265,59:POKE53272,24:REM GRAPHIK EIN/ BILDSCHIRMANFANG=8192
35 POKE49155,0:POKE49156,32:REM ROUTINE DEN BILSCHIRMANFANG MITTEILEN (8192)
37 POKE49157,AS:POKE49158,0:POKE49159,96:REM ZEICHENNR.,ZEICHENGENERATORANFANG
40 POKE49152,X:POKE49153,A:POKE49154,Y:REM BILDPUNKT VORBESETZEN
60 SYS49408 :REM ROUTINENANSPRUNG
70 GETA$:IFA$=""THEN70:REM WARTESCHLEIFE
80 PRINT"□":POKE53272,21:POKE53265,27:GOTO15:REM NEUE BEDINGUNGEN
```

```
                XX      EQU $C000       X-KOORDINATE
                AA      EQU $C001       RE-/LI BILDHAELFTE
                YY      EQU $C002       Y-KOORDINATE
                ADL     EQU $C003
                ADH     EQU $C004       ANFANGSADR. GRAPHIKBILDSCHIRM
                ASCII   EQU $C005       BILDSCHIRMCODE DES ZEICHENS
                ZGENL   EQU $C006
                ZGENH   EQU $C007       ANFANGSADR. ZEICHENSATZ
                ZWI1    EQU $C008
                ZWI2    EQU $C009       Y-REST
                PART1   EQU $C030       ZWISCHENSPEICHER1
                PART2   EQU $C038          - " -        2


                        ORG $C010

C010: 804020 TAB        DFB 128,64,32,16,8,4,2,1
C013: 100804
C016: 0201


                ************************************
                *                                  *
                * ROUTINE ZUR AUSGABE EINES        *
                * ZEICHENS AUF DEM GRAPHIKBILD-    *
                * SCHIRM                           *
                *                                  *
                ************************************


                        ORG $C100


C100: AD03C0            LDA ADL
C103: 8562              STA 98          ADRESSE GRAPHIK-RAM
C105: AD04C0            LDA ADH         NACH ZEROPAGE
C108: 8563              STA 99
C10A: AD02C0 MARK4      LDA YY
C10D: 38                SEC
C10E: E908              SBC #8
C110: 8D02C0            STA YY
C113: 9015              BCC MARK1
C115: A200              LDX #0          SCHLEIFE, UM DIE LAGE DER
C117: A562   MARK3      LDA 98          Y-KOORDINATE IM GRAPHIK-RAM
C119: 18                CLC             ZU ERMITTELN
C11A: 6928              ADC #40
C11C: 8562              STA 98
C11E: 9002              BCC MARK2
C120: E663              INC 99
C122: E8     MARK2      INX
C123: E008              CPX #8
C125: D0F0              BNE MARK3
C127: 4C0AC1            JMP MARK4
C12A: 6908   MARK1      ADC #8
C12C: 8D09C0            STA ZWI2
C12F: AD00C0            LDA XX
C132: 4A                LSR
C133: 4A                LSR
C134: 4A                LSR
```

```
C135: 0E35C1          ASL             *X:=INT(X)
C138: 0A              ASL
C139: 0A              ASL
C13A: 18              CLC
C13B: 6562            ADC 98
C13D: 8562            STA 98
C13F: 9002            BCC MARK6
C141: E663            INC 99          ADRESSE ERMITTELT!
C143: AD01C0 MARK6    LDA AA          RECHTE BILDHAELFTE?
C146: F002            BEQ MARK7
C148: E663            INC 99          DANN PLUS 256

C14A: AD06C0 MARK7    LDA ZGENL
C14D: 8564            STA 100         ANFANGSADRESSE DES ZEICHEN-
C14F: AD07C0          LDA ZGENH       SATZES NACH ZEROPAGE
C152: 8565            STA 101
C154: A900            LDA #0
C156: 8D08C0          STA ZWI1
C159: AD05C0          LDA ASCII
C15C: 0E5CC1          ASL             *ZEICHENNUMMER * 2
C15F: 900C            BCC MARK8       NUMMER < 128
C161: EE08C0          INC ZWI1
C164: EE08C0          INC ZWI1
C167: EE08C0          INC ZWI1
C16A: EE08C0          INC ZWI1
C16D: 0E6DC1 MARK8    ASL             *4
C170: 9006            BCC MARK9
C172: EE08C0          INC ZWI1
C175: EE08C0          INC ZWI1
C178: 0E78C1 MARK9    ASL             *8
C17B: 9003            BCC MARK10
C17D: EE08C0          INC ZWI1
C180: 18      MARK10  CLC
C181: 6564            ADC 100         OFFSET+ANFANGSADR.
C183: 8564            STA 100         ERGIBT ABSOLUTADR. DES
C185: 9002            BCC MARK11      BITMUSTERS
C187: E665            INC 101
C189: AD08C0 MARK11   LDA ZWI1
C18C: 18              CLC
C18D: 6565            ADC 101
C18F: 8565            STA 101
C191: A007            LDY #7
C193: B164    MARK12  LDA (100),Y
C195: 9930C0          STA PART1,Y     BITMUSTER ZWISCHENSPEICHERN
C198: 88              DEY
C199: 10F8            BPL MARK12

C19B: A007            LDY #7
C19D: A900    MARK15  LDA #0
C19F: 9938C0          STA PART2,Y
C1A2: 88              DEY
C1A3: 10F8            BPL MARK15

C1A5: A200            LDX #0
C1A7: AD00C0 MARK14   LDA XX
C1AA: 2907            AND #7
C1AC: A8              TAY
C1AD: F018            BEQ MARK22
C1AF: BD30C0 MARK13   LDA PART1,X
C1B2: 4EB2C1          LSR             *BITMUSTER GEMAESS X-KOORDINATE
C1B5: 9D30C0          STA PART1,X     VERSCHIEBEN
```

```
C1B8: BD38C0          LDA PART2,X
C1BB: 6A              ROR
C1BC: 9D38C0          STA PART2,X       VERSCHIEBUNG IN 2. ZW.-SPEICHER
C1BF: 88              DEY
C1C0: D0ED            BNE MARK13
C1C2: E8              INX
C1C3: E008            CPX #8            ALLE BYTES GESHIFTET?
C1C5: D0E0            BNE MARK14
C1C7: A200    MARK22  LDX #0
C1C9: AC09C0          LDY ZWI2
C1CC: A563    MARK16  LDA 99
C1CE: 8565            STA 101
C1D0: A562            LDA 98            SETZEN ZP FUER EINBLENDUNG
C1D2: 18              CLC
C1D3: 6908            ADC #8
C1D5: 8564            STA 100
C1D7: 9002            BCC MARK18
C1D9: E665            INC 101
C1DB: B162    MARK18  LDA (98),Y
C1DD: 1D30C0          ORA PART1,X       EINBLENDUNG 1. SHIFT-TEIL
C1E0: 9162            STA (98),Y
C1E2: B164            LDA (100),Y
C1E4: 1D38C0          ORA PART2,X       EINBLENDUNG 2. SHIFT-TEIL
C1E7: 9164            STA (100),Y
C1E9: E8              INX
C1EA: C8              INY
C1EB: C008            CPY #8            ZEILE BEENDET?
C1ED: D0DD            BNE MARK16
C1EF: E008            CPX #8            8 BYTE EINGEBLENDET?
C1F1: F030            BEQ MARK17
C1F3: A562            LDA 98
C1F5: 18              CLC
C1F6: 6940            ADC #64
C1F8: 8562            STA 98            NEIN! DANN 1 ZEILE RUNTER
C1FA: 9002            BCC MARK19
C1FC: E663            INC 99
C1FE: E663    MARK19  INC 99
C200: A564            LDA 100
C202: 18              CLC
C203: 6940            ADC #64
C205: 8564            STA 100
C207: 9002            BCC MARK20
C209: E665            INC 101
C20B: E665    MARK20  INC 101
C20D: A000            LDY #0
C20F: B162    MARK21  LDA (98),Y        EINBLENDEN REST IN NAECHSTE
C211: 1D30C0          ORA PART1,X       ZEILE
C214: 9162            STA (98),Y
C216: B164            LDA (100),Y
C218: 1D38C0          ORA PART2,X
C21B: 9164            STA (100),Y
C21D: E8              INX
C21E: C8              INY
C21F: E008            CPX #8            ALLE BYTES EINGEBLENDET?
C221: D0EC            BNE MARK21
C223: 60      MARK17  RTS               FERTIG!
PHYSICAL ENDADDRESS: $C224
```

*** NO WARNINGS

XX	$C000		AA	$C001
YY	$C002		ADL	$C003
ADH	$C004		ASCII	$C005
ZGENL	$C006		ZGENH	$C007
ZWI1	$C008		ZWI2	$C009
PART1	$C030		PART2	$C038
TAB	$C010	UNUSED	MARK4	$C10A
MARK3	$C117		MARK2	$C122
MARK1	$C12A		MARK6	$C143
MARK7	$C14A		MARK8	$C16D
MARK9	$C178		MARK10	$C180
MARK11	$C189		MARK12	$C193
MARK15	$C19D		MARK14	$C1A7
MARK13	$C1AF		MARK22	$C1C7
MARK16	$C1CC		MARK18	$C1DB
MARK19	$C1FE		MARK20	$C20B
MARK21	$C20F		MARK17	$C223

Notizen

Programmstop

Diese Unterroutine ist interruptgesteuert, d.h., sie wird bei jedem Hardwareinterrupt durchlaufen.

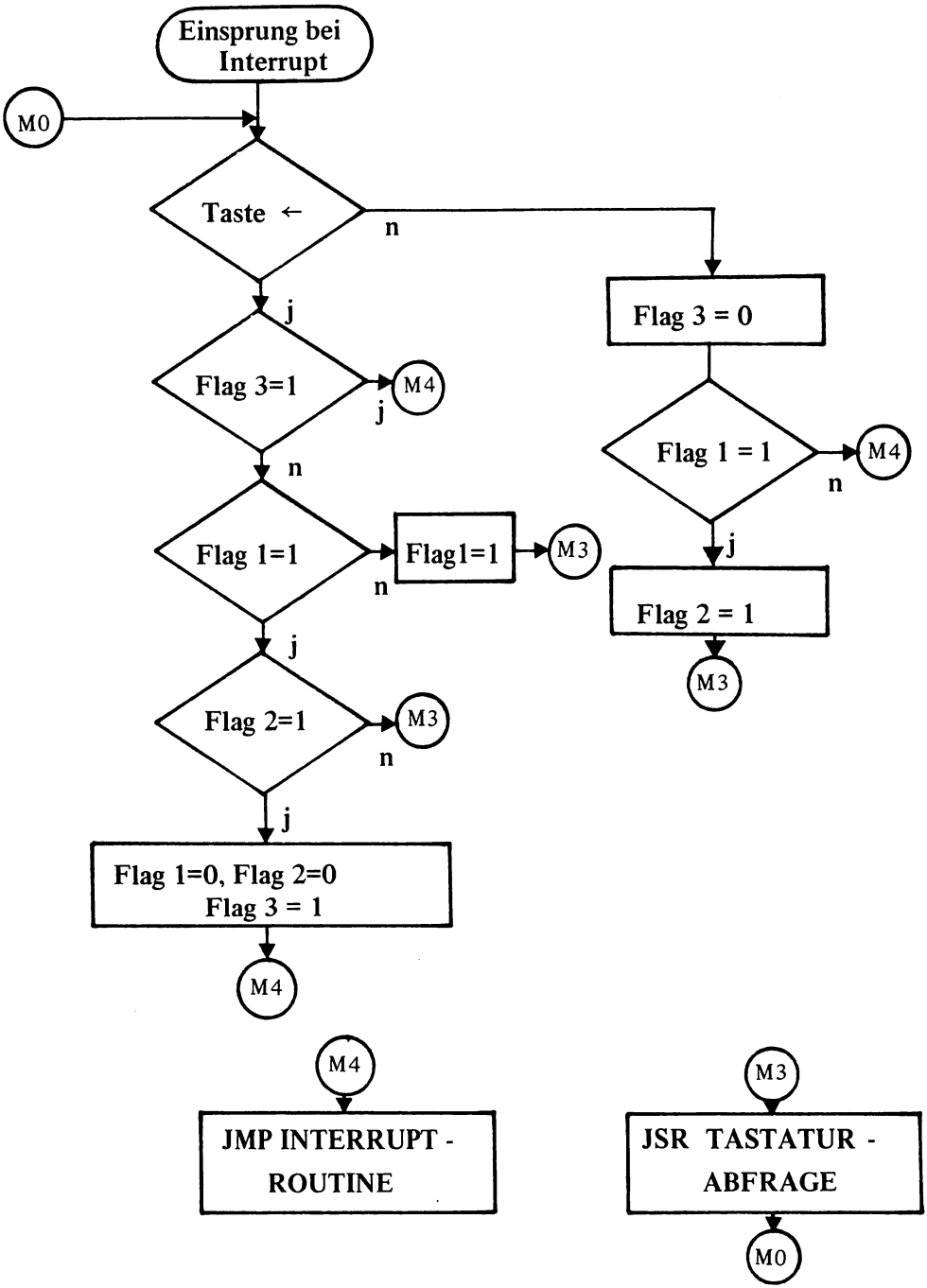
Die Funktionsweise ist folgende:

Wird die Betätigung der Taste '<-' erkannt, dann verzweigt das Programm in eine Schleife, in der nur noch die Tastaturmatrix abgefragt wird. Dadurch wird jedes laufende Programm (auch Maschinenprogramm) gestoppt. Die Schleife wird durch nochmaliges Drücken der Taste '<-' verlassen. Dadurch wird ein unterbrochenes Programm festgesetzt.

Der Sinn der Sache:

Sicher haben Sie sich beim Listen von BASIC-Programmen schon oft geärgert, daß das Listing nur verlangsamt oder abgebrochen werden kann. Mit dieser Maschinenroutine können Sie BASIC-Listings stoppen und auf Tastendruck wieder weiterführen. Ebenso können Sie Ihr Programm an interessanten Stellen anhalten und dann wieder fortsetzen. Die Routine funktioniert bei allen Rechnertätigkeiten, in denen das Interruptflag (SEI) nicht gesetzt ist.

Hier das Ablaufdiagramm der Stopproutine:



```
                FLAG1    EQU $C025
                FLAG2    EQU $C026
                FLAG3    EQU $C027

                         ORG $C000

C000: 78                 SEI                 INTERRUPT SPERREN
C001: A930               LDA #$30
C003: 8D1403             STA 788
C006: A9C0               LDA #$C0            IR-VEKTOR UMLENKEN
C008: 8D1503             STA 789
C00B: A900               LDA #0
C00D: 8D25C0             STA FLAG1
C010: 8D26C0             STA FLAG2
C013: 8D27C0             STA FLAG3
C016: 58                 CLI                 IR FREIGEBEN
C017: 60                 RTS

                         ORG $C030

C030: A5CB    M0         LDA 203             TASTE='_' ?
C032: C939               CMP #57
C034: D023               BNE M1              NEIN
C036: AD27C0             LDA FLAG3           TASTE NOCH GEDRUECKT
C039: D017               BNE M4              JA
C03B: AD25C0             LDA FLAG1           INZWISCHEN ANDERE TASTE?
C03E: F02B               BEQ M2              JA
C040: AD26C0             LDA FLAG2           '_' ZWEITES MAL?
C043: F02B               BEQ M3              NEIN
C045: A900               LDA #0              FLAGS RUECKSETZEN
C047: 8D25C0             STA FLAG1           FUER NEUE ABFRAGE
C04A: 8D26C0             STA FLAG2
C04D: A901               LDA #1
C04F: 8D27C0             STA FLAG3
C052: A900    M4         LDA #0
C054: 85C6               STA 198
C056: 4C31EA             JMP $EA31           NORMALE INTERUPTBEHANDLUNG
C059: A900    M1         LDA #0              TASTE '_' LOSGELASSEN
C05B: 8D27C0             STA FLAG3
C05E: AD25C0             LDA FLAG1
C061: F0EF               BEQ M4
C063: A901               LDA #1
C065: 8D26C0             STA FLAG2
C068: 4C70C0             JMP M3
C06B: A901    M2         LDA #1
C06D: 8D25C0             STA FLAG1
C070: 78      M3         SEI
C071: 2087EA             JSR $EA87           TASTATURABFRAGE
C074: 4C30C0             JMP M0
PHYSICAL ENDADDRESS: $C077


*** NO WARNINGS

FLAG1            $C025          FLAG2            $C026
FLAG3            $C027          M0               $C030
M4               $C052          M1               $C059
M2               $C06B          M3               $C070
```

Uhr in Maschinensprache

Wollen Sie schon lange eine sekundengenaue Uhr, die Ihnen unabhängig von Ihrem Programm die Zeit anzeigt?

Das folgende Maschinenprogramm stellt Ihnen eine Uhr zur Verfügung, die Sie nach Belieben in den Bildschirm einblenden können.

Das Programm ist interruptgesteuert, d. h. , die Zeitangabe wird jede 1/60 Sek. durch den Hardwareinterrupt auf den aktuellen Stand gebracht, und falls gewünscht, in der rechten oberen Ecke des Bildschirms eingeblendet. Durch die Interruptsteuerung entfällt eine zeitmäßige Belastung Ihrer eigenen Programme. Als Zeitgeber wurde die Echtzeituhr im CIA 6526 verwendet, die von der Netzfrequenz (50 Hz) gesteuert wird und dadurch im Vergleich zu der vom Betriebssystem versorgten Uhr 'TI$' eine sehr gute Genauigkeit aufweist.

Stellen und Starten der Uhr:

- Laden Sie die Speicherstellen 49152 - 49159 mit den Stunden, Minuten und Sekunden, z.B. POKE 49152,17: POKE 49153,34: POKE 49154,0 bedeutet 17 h 34 min 0 sek.

- In Zelle 49155 können Sie festlegen, ob die Uhrzeit eingeblendet (POKE 49155,1) oder ausgeblendet (POKE 49155,0) wird.

Die Uhr wird gestartet mit SYS 49168.
Dadurch wird der Hardwareinterruptvektor
(788,789 dez) umgelenkt und somit bei
jedem Interrupt zuerst die Uhrzeitroutine
angesprungen. Außerdem wird die Echtzeituhr
im CIA 6526 durch diese Routine gestellt.

Das Stellen der Uhr und das Ein- und
Ausblenden der Zeitangabe ist zu jedem
Zeitpunkt möglich.

Die Uhr ist eine 24-Sekunden-Uhr, die
Stundenangabe geht aber nur bis 12 Uhr.
Für Vormittags steht hinter der Zeitangabe
AM (Ante Meridiem) für Nachmittags
steht PM (Post Meridiem).

```
HH      EQU $C000       STUNDEN
MM      EQU $C001       MINUTEN
SS      EQU $C002       SEKUNDEN
EA      EQU $C003       EIN-/AUSBLENDEN


*************************************
*                                   *
*     INTERRUPTVEKTOR SETZEN        *
*     UHR STELLEN                   *
*                                   *
*************************************

                ORG $C010

C010: 78        SEI             INTERRUPT AUS
C011: AD0EDC    LDA 56334       50 HZ EINSCHALTEN
C014: 0980      ORA #128
C016: 8D0EDC    STA 56334
C019: AD0FDC    LDA 56335       STELLMODUS EIN
C01C: 297F      AND #127
C01E: 8D0FDC    STA 56335
C021: AD02C0    LDA SS
C024: C93C      CMP #60         SEK EINGABE >= 60
C026: B05E      BCS FEHLER
C028: AD01C0    LDA MM
```

```
C02B: C93C            CMP #60           MIN EINGABE >= 60
C02D: B057            BCS FEHLER
C02F: AD00C0          LDA HH
C032: C918            CMP #24           STD EINGABE >= 24
C034: B050            BCS FEHLER
C036: C90C            CMP #12
C038: 9006            BCC MARK1
C03A: 18              CLC
C03B: 6974            ADC #116          KORREKTUR STD FUER BCD-FORMAT
C03D: 8D00C0          STA HH
C040: A900    MARK1   LDA #$00          IR-VEKTOR UMLEITEN
C042: 8D1403          STA 788
C045: A9C1            LDA #$C1
C047: 8D1503          STA 789
C04A: AD01C0          LDA MM            MIN IN BCD-FORMAT
C04D: 2091C1          JSR BCD
C050: 8D01C0          STA MM
C053: AD02C0          LDA SS
C056: 2091C1          JSR BCD           SEK IN BCD-FORMAT
C059: 8D02C0          STA SS
C05C: 18              CLC
C05D: F8              SED               DEZ EIN
C05E: AD00C0          LDA HH
C061: 6900            ADC #0            KORR. STD FUER BCD
C063: 8D0BDC          STA 56331         STD.-REG. STELLEN
C066: AD01C0          LDA MM
C069: 18              CLC
C06A: 6900            ADC #0            MIN IN BCD
C06C: 8D0ADC          STA 56330         MIN.-REG. STELLEN
C06F: AD02C0          LDA SS
C072: 18              CLC
C073: 6900            ADC #0            SEK IN BCD
C075: 8D09DC          STA 56329         SEK.-REG. STELLEN
C078: A900            LDA #0
C07A: 8D08DC          STA 56328         1/10 SEK.-REG=0
C07D: D8              CLD               DEZ AUS
C07E: A920            LDA #32
C080: 8D2504          STA 1061          FRAGEZEICHEN AUSBLENDEN
C083: 4C90C0          JMP MARK2
C086: A93F    FEHLER  LDA #63
C088: 8D2504          STA 1061          FRAGEZEICHEN EIN
C08B: A901            LDA #1
C08D: 8D23D8          STA 55331         FARBE WEISS
C090: 58      MARK2   CLI
C091: 60              RTS
```

```
                ************************************
                *                                  *
                *       HAUPTROUTINE UHR           *
                *                                  *
                ************************************

                        ORG $C100

C100: AD03C0            LDA EA            ANZEIGE EIN/AUS?
C103: D003              BNE MARK7
C105: 4C31EA            JMP $EA31         NORMALER INTERRUPT
C108: AD0BDC  MARK7     LDA 56331         STD
C10B: 297F              AND #127          AM/FM AUSBLENDEN
C10D: 2085C1            JSR SHIFT
```

```
C110: 8D1D04          STA 1053          AUSGABE 10ER STD
C113: AD0BDC          LDA 56331
C116: 290F            AND #$0F
C118: 18              CLC
C119: 6930            ADC #48
C11B: 8D1E04          STA 1054          AUSGABE 1ER STD
C11E: A92E            LDA #46
C120: 8D1F04          STA 1055
C123: AD0ADC          LDA 56330
C126: 2085C1          JSR SHIFT
C129: 8D2004          STA 1056          AUSGABE 10ER MIN
C12C: AD0ADC          LDA 56330
C12F: 290F            AND #$0F
C131: 18              CLC
C132: 6930            ADC #48
C134: 8D2104          STA 1057          AUSGABE 1ER MIN
C137: A92E            LDA #46
C139: 8D2204          STA 1058
C13C: AD09DC          LDA 56329
C13F: 2085C1          JSR SHIFT
C142: 8D2304          STA 1059          AUSGABE 10ER SEK
C145: AD09DC          LDA 56329
C148: 290F            AND #$0F
C14A: 18              CLC
C14B: 6930            ADC #48
C14D: 8D2404          STA 1060          AUSGABE 1ER SEK
C150: AD0BDC          LDA 56331
C153: 2980            AND #128          AM/PM?
C155: F005            BEQ MARK3
C157: A910            LDA #16           PM!
C159: 4C6DC1          JMP MARK4
C15C: AD0BDC MARK3    LDA 56331
C15F: C912            CMP #18           12 UHR AM = MITTERNACHT
C161: D008            BNE MARK6
C163: A900            LDA #0            DANN 0 UHR AM
C165: 8D0BDC          STA 56331
C168: 8D08DC          STA 56328
C16B: A901    MARK6   LDA #1            AM!
C16D: 8D2604 MARK4    STA 1062
C170: AD08DC          LDA 56328         UHR FREIGEBEN
C173: A90D            LDA #13           AUSGABE 'M'
C175: 8D2704          STA 1063
C178: A20B            LDX #11
C17A: A901            LDA #1
C17C: 9D1CD8 MARK5    STA 55324,X       FARBE WEISS
C17F: CA              DEX
C180: D0FA            BNE MARK5
C182: 4C31EA          JMP $EA31

C185: 4E85C1 SHIFT    LSR               *RECHTSBUENDIG UND IN
C188: 4E88C1          LSR               *ASCII-CODE UMWANDELN
C18B: 4A              LSR
C18C: 4A              LSR
C18D: 18              CLC
C18E: 6930            ADC #48
C190: 60              RTS

C191: C938    BCD     CMP #56           KORREKTUR DEZ NACH BCD
C193: 9003            BCC KONO
C195: 18              CLC
C196: 6906            ADC #6
```

```
C198: C92E    KON0      CMP #46
C19A: 9003              BCC KON1
C19C: 18                CLC
C19D: 6906              ADC #6
C19F: C924    KON1      CMP #36
C1A1: 9003              BCC KON2
C1A3: 18                CLC
C1A4: 6906              ADC #6
C1A6: C91A    KON2      CMP #26
C1A8: 9003              BCC KON3
```

```
C1AA: 18                CLC
C1AB: 6906              ADC #6
C1AD: C910    KON3      CMP #16
C1AF: 9003              BCC END
C1B1: 18                CLC
C1B2: 6906              ADC #6
C1B4: 60      END       RTS
PHYSICAL ENDADDRESS: $C1B5

*** NO WARNINGS
```

HH	$C000	MM	$C001
SS	$C002	EA	$C003
MARK1	$C040	FEHLER	$C086
MARK2	$C090	MARK7	$C108
MARK3	$C15C	MARK6	$C16B
MARK4	$C16D	MARK5	$C17C
SHIFT	$C185	BCD	$C191
KON0	$C198	KON1	$C19F
KON2	$C1A6	KON3	$C1AD
END	$C1B4		

Ausgabe einer Hexzahl

Das Programm Hexausgabe dient dazu, den Wert einer Speicherzelle in Hexadezimal auf dem Bildschirm auszugeben.

Der Wert der Speicherzelle wird in Adresse C002 hex (49154 dez) zwischengespeichert.

Die Zeilen- und Spaltenposition, an der die Ausgabe erfolgen soll, werden in Zelle C000 und C001 abgelegt.

Die Farbe der Zeichen schließlich wird in Zelle C003 bestimmt.

Beispiel:

```
C000 = 10    bedeutet Ausgabe in Zeile 10
C001 = 15    bedeutet Ausgabe in Spalte 15
C002 = 159   Wert, der umgewandelt wird
C004 = 2     Farbe = rot
```

Bei diesen Angaben wird die Hexadezimalzahl 'A9' in roter Farbe in Zeile 10 ab Spalte 15 am Bildschirm ausgegeben.

```
ZEILE    EQU $C000          ZEILENNUMMER
SPALTE   EQU $C001          SPALTENNUMMER
ZAHL     EQU $C002          WERT DER ZAHL
FARBE    EQU $C003          FARBE DES ZEICHENS
ZEICHEN  EQU $C004          ZWISCHENZELLE
CURSOR   EQU $E50A          SUBROUTINE CURSOR SETZEN
FARBRAM  EQU $EA24          SUBROUTINE FARBRAM ERMITTELN
AUSGABE  EQU $EA1C          AUSGABE ZEICHEN + FARBE



*************************************
*                                   *
* AUSGABE EINER HEXZAHL             *
*                                   *
*************************************

         ORG $C010

C010: AD02C0          LDA ZAHL
C013: 29F0            AND #$F0        RECHTE HAELFTE AUSBLENDEN
C015: 4A              LSR
C016: 4A              LSR
C017: 4A              LSR
C018: 4E18C0          LSR             *RECHTSBUENDIG
C01B: 205FC0          JSR DISG        IN BILDSCHIRMCODE UMWANDELN
C01E: 8D04C0          STA ZEICHEN
C021: 2048C0          JSR AUS         ZEICHEN AUSGEBEN
C024: AD01C0          LDA SPALTE
C027: C927            CMP #39         ZEILENENDE
C029: D00B            BNE INCSPA      NEIN!
C02B: A900            LDA #0
C02D: 8D01C0          STA SPALTE      NEUE ZEILE
C030: EE00C0          INC ZEILE
C033: 4C39C0          JMP WEITER
C036: EE01C0 INCSPA   INC SPALTE      EINE SPALTE RECHTS
C039: AD02C0 WEITER   LDA ZAHL
C03C: 290F            AND #$0F        LINKE HAELFTE AUSBLENDEN
C03E: 205FC0          JSR DISG        IN BILDSCHIRMCODE UMWANDELN
C041: 8D04C0          STA ZEICHEN
C044: 2048C0          JSR AUS         AUSGABE
C047: 60              RTS             FERTIG!



C048: AE00C0 AUS      LDX ZEILE       ZEILE IN X-REG.
C04B: AC01C0          LDY SPALTE      SPALTE IN Y-REG.
C04E: 18              CLC
C04F: 200AE5          JSR CURSOR      CURSOR SETZEN
C052: 2024EA          JSR FARBRAM     ADR. FARBRAM BERECHNEN
C055: AD04C0          LDA ZEICHEN     ZEICHEN IN AKKU
C058: AE03C0          LDX FARBE       FARBE IN X-REG.
C05B: 201CEA          JSR AUSGABE     ZEICHEN MIT FARBE AUSGEBEN
C05E: 60              RTS             FERTIG!
C05F: C90A   DISG     CMP #10         ZAHL >= 10 DEZ?
C061: 9004            BCC DIGIT
C063: 38              SEC             ZAHL >= 10
```

```
C064: E909              SBC #9              BUCHSTABEN A-F
C066: 60                RTS
C067: 18      DIGIT     CLC                 ZAHL < 10
C068: 6930              ADC #48             ZIFFERN 0-9
C06A: 60                RTS
PHYSICAL ENDADDRESS: $C06B

*** NO WARNINGS

ZEILE               $C000                 SPALTE              $C001
ZAHL                $C002                 FARBE               $C003
ZEICHEN             $C004                 CURSOR              $E50A
FARBRAM             $EA24                 AUSGABE             $EA1C
INCSPA              $C036                 WEITER              $C039
AUS                 $C048                 DISG                $C05F
DIGIT               $C067
```

NOTIZEN

Ausgabe eines Zeichens auf dem Bildschirm

Wollen Sie in Maschinensprache ein Zeichen auf dem Bildschirm ausgeben, dann haben Sie im Wesentlichen zwei Möglichkeiten:

1. Sie berechnen die Bildschirmposition im VIDEO-RAM und setzen an diese Stelle den Bildschirmcode des Zeichens.
   Danach errechnen Sie die Position im Farb-RAM und setzen an den Punkt die gewünschte Farbe.

oder:

2. Sie verwenden eine ROM-Routine, die Ihnen aufgrund der eingegebenen Zeile und Spalte den Cursor auf die gewünschte Position setzt (URSOR).
   Danach wird eine Routine benutzt, die aufgrund der Cursorstellung die Position im Farb-RAM berechnet (FARBRAM). Schließlich wird mit einer dritten Routine das Zeichen samt Farbe ausgegeben (AUSGABE).
   Diese zwei Routinen sollen hier gegenübergestellt werden.

Entscheiden Sie, welche der beiden Routinen fuer Ihr Problem geeignet ist.

Zur Beachtung:

In der Zelle 'ZEICHEN' muß der Bildschirmcode des Zeichens (nicht des ASCII-Codes) stehen.

z.B. Zeichen = "D" -> 'ZEICHEN':=4 (nicht 68).

Als Farben sind die Werte 0 - 15 zulässig.

Der Befehl 'CLC' in Routine 2 bedeutet, daß der Cursor gesetzt wird. Will man die momentane Position des Cursors erhalten, dann muß vor den Ansprung der Routine 'CURSOR' der Befehl 'SEC' (Set Carry) gesetzt werden.

Im X-Register wird dann die Zeilennummer und im Y-Register die Spaltennummer der momentanen Cursor-Position übergangen.

```
ZEILE    EQU $C000
SPALTE   EQU $C001
ZEICHEN  EQU $C002
FARBE    EQU $C003
CURSOR   EQU $E50A
FARBRAM  EQU $EA24
AUSGABE  EQU $EA1C


*************************************
*                                   *
* ZEICHENAUSGABE AUF DEN BILDSCHIRM *
*                                   *
*************************************

         ORG $C010


**********
*
* ROUTINE1
*
**********


C010: A900           LDA #0
C012: 8562           STA 98          BILDSCHIRMANFANG LADEN
C014: A904           LDA #4          HIER 1024 DEZ
C016: 8563           STA 99
C018: AE00C0         LDX ZEILE       X-REG. MIT ZEILE LADEN
C01B: CA     MARK3   DEX
C01C: F00E           BEQ MARK1       ALLE ZEILEN ABGEARBEITET
C01E: A562           LDA 98
C020: 18             CLC
C021: 6928           ADC #40         1 ZEILE WEITER
C023: 8562           STA 98
C025: 9002           BCC MARK2
C027: E663           INC 99          SEITENUEBERSCHREITUNG
C029: 4C1BC0 MARK2   JMP MARK3
C02C: A562   MARK1   LDA 98
C02E: 18             CLC
C02F: 6D01C0         ADC SPALTE      SPALTE ADDIEREN
C032: 8562           STA 98
C034: 9002           BCC MARK4
C036: E663           INC 99          SEITENUEBERSCHREITUNG
C038: AD02C0 MARK4   LDA ZEICHEN
C03B: A200           LDx #0
C03D: 8162           STA (98,X)      ZEICHEN AUSGEBEN
C03F: A563           LDA 99
C041: 18             CLC
C042: 69D4           ADC #212        FARBRAM
C044: 8563           STA 99
C046: AD03C0         LDA FARBE
C049: 8162           STA (98,X)      FARBE AUSGEBEN
C04B: 60             RTS
```

```
            **********
            *
            * ROUTINE2
            *
            **********


C04C: AE00C0          LDX ZEILE         ZEILE IN X-REG.
C04F: AC01C0          LDY SPALTE        SPALTE IN Y-REG.
C052: 18              CLC
C053: 200AE5          JSR CURSOR        CURSOR SETZEN
C056: 2024EA          JSR FARBRAM       ADR. FARBRAM BERECHNEN
C059: AD02C0          LDA ZEICHEN       ZEICHEN IN AKKU
C05C: AE03C0          LDX FARBE         FARBE IN X-REG.
C05F: 201CEA          JSR AUSGABE       ZEICHEN MIT FARBE AUSGEBEN
C062: 60              RTS               FERTIG!
PHYSICAL ENDADDRESS: $C063

*** NO WARNINGS

ZEILE             $C000                 SPALTE            $C001
ZEICHEN           $C002                 FARBE             $C003
CURSOR            $E50A                 FARBRAM           $EA24
AUSGABE           $EA1C                 MARK3             $C01B
MARK2             $C029                 MARK1             $C02C
MARK4             $C038
```

Bildpunkt zeichnen in Maschinensprache

Das Zeichnen eines Bildpunktes im Graphikmodus bei gegebenen X- und Y-Koordinaten ist in BASIC sehr aufwendig und zeitraubend und kann in Maschinensprache weitaus schneller erledigt werden.

Aus diesem Grund wurde nachfolgendes Maschinenprogramm erstellt, das diese Routinearbeit übernimmt.

Das Programm beginnt ab Adresse C100 hex (49408 dez).

In Zelle E000 hex (49152) wird der Wert der X-Koordinate abgelegt, in Zelle C002 der der Y-Koordinate. In Zelle C001 wird dem Programm mitgeteilt, ob der Bildpunkt im linken Bildschirmteil (C001 = 0) oder im rechten Bildschirmteil liegt (C001 = 1).

In Adresse C004 und C005 muß die Anfangsadresse des 8k großen Graphikbildschirms abgelegt werden.

Die Verwendung der Maschinenroutine in Verbindung mit BASIC demonstriert das nachfolgende, kleine BASIC-Programm. das eine Diagonale vom linken oberen Bildschirmrand zum rechten unteren Rand zeichnet. Die Schnelligkeit der Maschinenunterroutine läßt sich jedoch nur in Verbindung mit einem Maschinenprogramm voll ausnutzen.

```
10 FORI=8192TO16192:POKEI,0:NEXT:REM GRAPHIKBILDSCHIRM LOESCHEN
20 POKE53265,59:POKE53272,24:REM GRAPHIK EIN/ BILDSCHIRMANFANG=8192
30 X=0:A=0:Y=0
35 POKE49155,0:POKE49156,32:REM ROUTINE DEN BILDSCHIRMANFANG MITTEILEN
   !(8192)
40 POKE49152,X:POKE49153,A:POKE49154,Y:REM BILDPUNKT ZEICHNEN
50 X=X+2:Y=Y+1:IFX=256THENX=0:A=1
60 SYS49408:GOTO40:REM ROUTINENAMENPUNG
```

```
                XX      EQU $C000       X-KOORDINATE
                AA      EQU $C001       RE-/LI BILDHAELFTE
                YY      EQU $C002       Y-KOORDINATE
                ADL     EQU $C003       ADR. LOW GRAPHIKANFANG
                ADH     EQU $C004       ADR. HIGH

                        ORG $C010

C010: 804020 TAB        DFB 128,64,32,16,8,4,2,1
C013: 100804
C016: 0201

                ****************************************
                *                                      *
                *       ZEICHNEN EINES PUNKTES         *
                *                                      *
                ****************************************

                        ORG $C100

C100: AD03C0            LDA ADL         ADRESSE NACH ZEROPAGE
C103: 8562              STA 98
C105: AD04C0            LDA ADH
C108: 8563              STA 99
C10A: AD02C0 MARK4      LDA YY
C10D: 38                SEC
C10E: E908              SBC #8
C110: 8D02C0            STA YY          SCHLEIFE FUER BERECHNUNG
C113: 9015              BCC MARK1       IN WELCHER REIHE
C115: A200              LDX #0          Y-KOORDINATE LIEGT
C117: A562   MARK3      LDA 98
C119: 18                CLC
C11A: 6928              ADC #40
C11C: 8562              STA 98
C11E: 9002              BCC MARK2
C120: E663              INC 99
C122: E8     MARK2      INX
C123: E008              CPX #8
C125: D0F0              BNE MARK3
C127: 4C0AC1            JMP MARK4
```

```
C12A: 6908    MARK1     ADC #8
C12C: 18                CLC
C12D: 6562              ADC 98
C12F: 8562              STA 98
C131: 9002              BCC MARK5
C133: E663              INC 99
C135: AD00C0  MARK5     LDA XX
C138: 4A                LSR
C139: 4A                LSR
C13A: 4A                LSR
C13B: 0E3BC1            ASL             *X=INT(X)
C13E: 0A                ASL
C13F: 0A                ASL
C140: 18                CLC
C141: 6562              ADC 98
C143: 8562              STA 98          POS. IN GRAPHIK-RAM FESTGELEGT
C145: 9002              BCC MARK6
C147: E663              INC 99
C149: AD01C0  MARK6     LDA AA          RECHTER TEIL?
C14C: F002              BEQ MARK7
C14E: E663              INC 99          DANN PLUS 256
C150: A200    MARK7     LDX #0
C152: AD00C0            LDA XX
C155: 2907              AND #7          WIEVIELTES BIT
C157: A8                TAY
C158: A162              LDA (98,X)
C15A: 1910C0            ORA TAB,Y       AUSGABE
C15D: 8162              STA (98,X)
C15F: 60                RTS
PHYSICAL ENDADDRESS: $C160

*** NO WARNINGS

XX              $C000           AA              $C001
YY              $C002           ADL             $C003
ADH             $C004           TAB             $C010
MARK4           $C10A           MARK3           $C117
MARK2           $C122           MARK1           $C12A
MARK5           $C135           MARK6           $C149
MARK7           $C150
```

NOTIZEN

Veränderbare Melodie in Maschinensprache

Dieses kleine Soundprogramm soll zur Demonstration dienen, wie man Melodien in Maschinensprache erstellen kann, und wie man die Klänge mit Hilfe von Filtereinstellung und Filterfrequenz verfremden kann.

In Adresse $C000 wird die Wellenform eingegeben (Rauschen = 129 dez, Rechteck = 65, Sägezahn = 33 und Dreieck = 17).

In Zelle $C001 können Sie entscheiden, ob die Melodie 'normal' oder verfremdet wiedergegeben wird.

Legen Sie eine Eins in $C000 ab, dann wird der Filter für Stimme 1 eingeschaltet und als Filtertyp der Bandpaß gewählt.

In der Verzögerungsschleife (nach dem Anspielen einer Note) wird die Grenzfrequenz kontinuierlich geändert. Dadurch entsteht ein sogenannter Wow-Effekt.

Das Programm ist als Subroutine geschrieben und kann von BASIC oder Maschinensprache aus angesprungen werden (Ansprungadresse C100 hex, 49408 dez).

```
                WAVE    EQU $C000       WELLENFORM
                WOW     EQU $C001       VERZERRUNG J/N
                XREG    EQU $C002
                VERZOE  EQU $C003
                VERFL   EQU $C004


                ************************************
                *                                  *
                * NOTEN:                           *
                * FREQUENZ LOW,FREQUENZ HIGH       *
                * VERZOEGERUNG                     *
                *                                  *
                ************************************


                        ORG $C010

C010: B40802 SOUND DFB 180,8,2,158,11,1,10,13,2,162,14,1,10,13,4,247,10,5
C013: 9E0B01
C016: 0A0D02
C019: A20E01
C01C: 0A0D04
C01F: F70A05
C022: B40802       DFB 180,8,2,158,11,1,10,13,2,162,14,1,10,13,5
C025: 9E0B01
C028: 0A0D02
C02B: A20E01
C02E: 0A0D05
C031: B40802       DFB 180,8,2,158,11,1,10,13,2,162,14,1,10,13,4,247,10,5
C034: 9E0B01
C037: 0A0D02
C03A: A20E01
C03D: 0A0D04
C040: F70A05
C043: F70A02       DFB 247,10,2,158,11,1,247,10,2,180,8,1,180,8,5
C046: 9E0B01
C049: F70A02
C04C: B40801
C04F: B40805

                ************************************
                *                                  *
                * ABSPIELROUTINE                   *
                *                                  *
                ************************************


                        ORG $C100

    C100: A200          LDX #0
    C102: 8E02C0        STX XREG



    C105: A918          LDA #24
    C107: 8D05D4        STA 54277       ANSCHLAG
    C10A: A9F0          LDA #240
    C10C: 8D06D4        STA 54278       HALTEN
    C10F: A9F0          LDA #240
    C111: 18            CLC
```

```
C112: 6D01C0          ADC WOW          FILTER STI1 EIN,FALLS WOW=1
C115: 8D17D4          STA 54295        RESONANZ,FILTER EIN
C118: A92F            LDA #47          FILTER=BANDPASS
C11A: 8D18D4          STA 54296        LAUTSTAERKE VOLL
C11D: A900            LDA #0
C11F: 8D04D4          STA 54276        WAVE AUS
C122: AD00C0          LDA WAVE
C125: 8D04D4          STA 54276        WAVE EIN
C128: AE02C0 EIN2     LDX XREG
C12B: BD10C0          LDA SOUND,X      NOTE LOW
C12E: 8D00D4          STA 54272        FREQUENZ LOW
C131: E8              INX
C132: BD10C0          LDA SOUND,X      NOTE HIGH
C135: 8D01D4          STA 54273        FREQUENZ HIGH
C138: E8              INX
C139: BD10C0          LDA SOUND,X      VERZOEGERUNG
C13C: 8D03C0          STA VERZOE
C13F: E8              INX
C140: E045            CPX #69          ENDE?
C142: D006            BNE EIN3         NEIN!
C144: A900            LDA #0
C146: 8D06D4          STA 54278        AUSKLINGEN
C149: 60              RTS              FERTIG!
C14A: 8E02C0 EIN3     STX XREG
C14D: 2053C1          JSR ROUTIN
C150: 4C28C1          JMP EIN2         NAECHSTE NOTE


C153: A900   ROUTIN    LDA #0           VERZOEGERUNGSSCHLEIFE UND
C155: 8D04C0          STA VERFL        AENDERUNG DER GRENZFREQUENZ
C158: A000   VER3     LDY #0
C15A: A2BE   VER2     LDX #190
C15C: CA     VER1     DEX
C15D: D0FD            BNE VER1
C15F: AD04C0          LDA VERFL
C162: D003            BNE VER4
C164: 8C16D4          STY 54294        GRENZFREQUENZ
C167: C8     VER4     INY
C168: D0F0            BNE VER2
C16A: A901            LDA #1           GR.FREQU. BEREITS DURCHLAUFEN
C16C: 8D04C0          STA VERFL
C16F: CE03C0          DEC VERZOE
C172: D0E4            BNE VER3
C174: 60              RTS
PHYSICAL ENDADDRESS: $C175

*** NO WARNINGS

WAVE          $C000          WOW           $C001
XREG          $C002          VERZOE        $C003
VERFL         $C004          SOUND         $C010
EIN2          $C128          EIN3          $C14A
ROUTIN        $C153          VER3          $C158
VER2          $C15A          VER1          $C15C
VER4          $C167
```

NOTIZEN

Bitmuster spiegeln

Wenn Sie in einem Programm z. B. spiegelverkehrte Buchstaben verwenden wollen, dann kommen Sie nicht umhin, die Bitmuster der einzelnen Bytes zu spiegeln.

Das Unterprogramm 'Byte spiegeln' erledigt diese Spiegelung für ein Byte.

Das Bitmuster (die Zahl) wird in Zelle C000 hex eingegeben. In der selben Zelle erhalten Sie den gespiegelten Wert zurück.

```
BYTE     EQU $C000
ZWIZEL   EQU $C001

*************************************
*                                   *
*    BYTE SPIEGELN                  *
*                                   *
*************************************

         ORG $C010

C010: A900           LDA #0
C012: 8D01C0         STA ZWIZEL       ZWISCHENZELLE=0
C015: A208           LDX #8           ZAEHLER=8
C017: AD00C0 MARK1   LDA BYTE
C01A: 4E1AC0         LSR              *RECHTES BIT IN CARRY
C01D: 8D00C0         STA BYTE         ABSPEICHERN
C020: AD01C0         LDA ZWIZEL
C023: 2E23C0         ROL              *CARRY RECHTS EINSCHIEBEN
C026: 8D01C0         STA ZWIZEL       ABSPEICHERN
C029: CA             DEX              *ZAEHLER ERNIEDRIGEN
C02A: D0EB           BNE MARK1        ZAEHLER NULL,DANN ENDE
C02C: AD01C0         LDA ZWIZEL       GESPIEGELTES BYTE
C02F: 8D00C0         STA BYTE         ZURUECKSCHREIBEN
C032: 60             RTS              *FERTIG!
PHYSICAL ENDADDRESS: $C033

*** NO WARNINGS

BYTE             $C000               ZWIZEL           $C001
MARK1            $C017
```

NOTIZEN

Verzögerungsschleife

Speziell bei Spielen in Maschinensprache ist es eine Notwendigkeit, Verzögerungsschleifen einzubauen, da die Abarbeitungsgeschwindigkeit so hoch ist, daß ein normaler Spielverlauf ohne Verzögerung nicht mehr möglich ist.

In der folgenden Routine kann die Verzögerungsdauer in der Zelle 'VERZ' (C000 hex) eingestellt werden. Dadurch kann ein Spiel beschleunigt oder verlangsamt werden.

Bei Verz = 0 wird die Unterroutine 254 mal, bei Verz = 255 wird sie 64516 mal durchlaufen.

Experimentieren Sie, um die für Sie optimale Verzögerungsdauer herauszufinden.

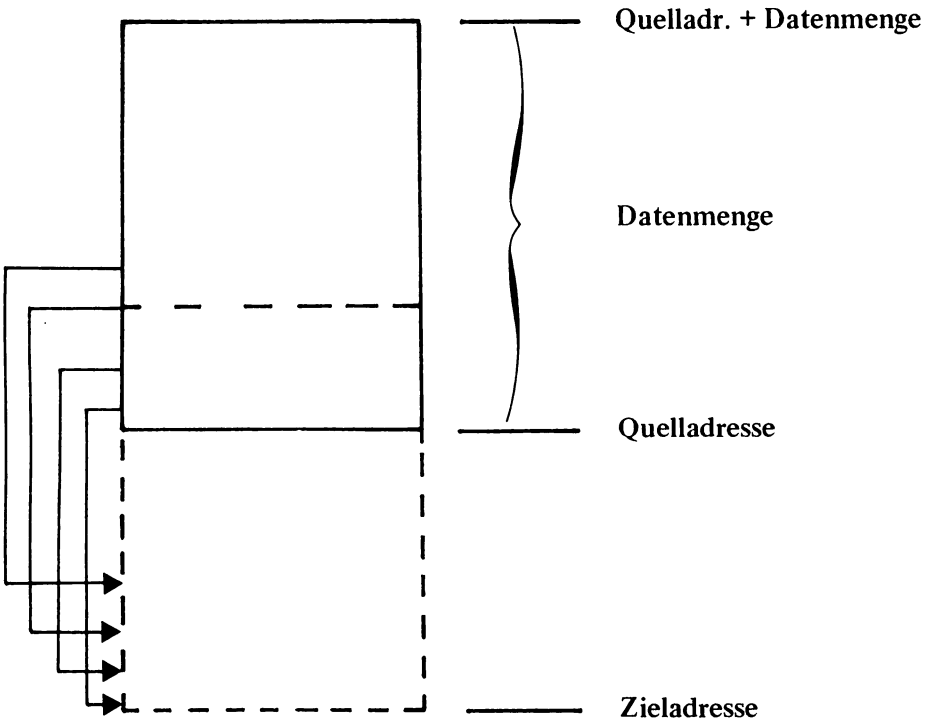
```
              VERZ      EQU $C000          VERZOEGERUNGSWERT


              ****************************
              *                          *
              *   VERZOEGERUNGSSCHLEIFE  *
              *                          *
              ****************************


                        ORG $C010

C010: AC00C0            LDY VERZ
C013: A2FF    MARK1     LDX #255
C015: CA      MARK2     DEX
C016: D0FD              BNE MARK2
C018: 88                DEY
C019: D0F8              BNE MARK1
C01B: 60                RTS
PHYSICAL ENDADDRESS: $C01C

*** NO WARNINGS

VERZ            $C000                   MARK1           $C013
MARK2           $C015
```

Blocktransfer

Das Unterprogramm Blocktransfer verschiebt eine beliebige Datenmenge von einer Quelladresse nach einer Zieladresse.

Für die Anordnung von Quell- und Zieladresse gibt es zwei Möglichkeiten, die beim Transfer berücksichtigt werden müssen.

1. QUELLADRESSE > ZIELADRESSE, d.h., die Daten werden im Speicher von oben nach unten verschoben. Den Ablauf des Transfers verdeutlicht das folgende Diagramm.



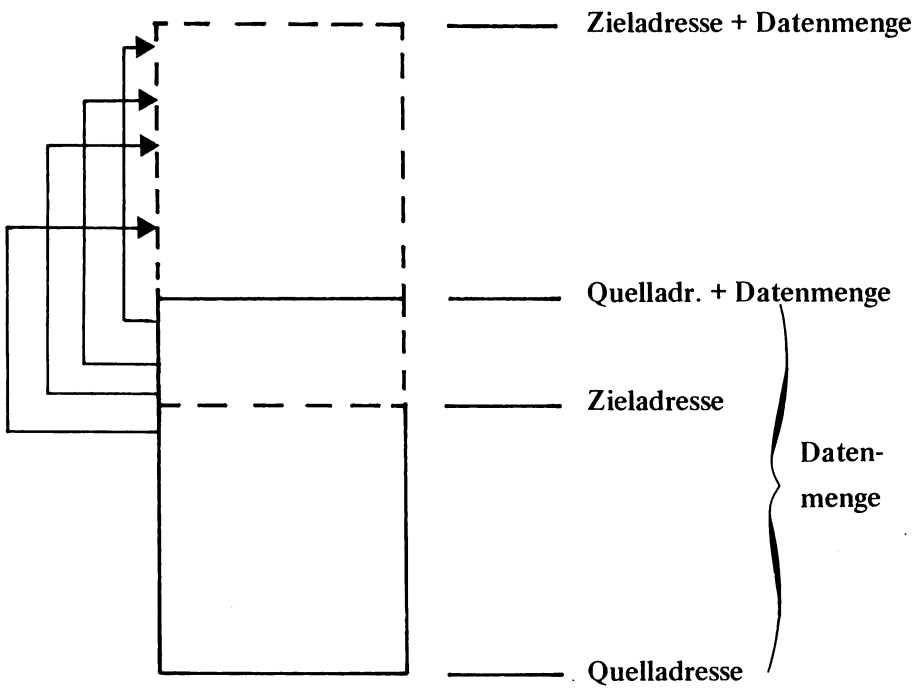
Die Verschiebung muß hierbei wie mit den Pfeilen angedeutet erfolgen:

```
Quelladresse   +    0        ->  Zieladresse  +  0

Quelladresse   +    1        ->  Zieladresse  +  1
    .              .                  .           .
    .              .                  .           .
    .              .                  .           .
Quelladresse   + Datenmenge  ->  Zieladresse + Datenmenge
```

Würden bei dieser Konstellation die Daten von oben her verschoben, dann würden bei Überlappung der Quell- und Zielblöcke die überlappten Adressen überschrieben werden, bevor sie transferiert werden können.

2. QUELLADRESSE < ZIELADRESSE

Verschiebung der Daten von unten nach oben. Der Transfer am Extrembeispiel der Überlappung:



Hier muß im Gegensatz zum vorangegangenen Beispiel die Verschiebung umgekehrt erfolgen, da sonst wieder Bereiche vor dem Transfer überschrieben würden.

```
Quelladresse  + Datenmenge     -> Zieladresse + Datenmenge
Quelladresse  +(Datenmenge-1)  -> Zieladresse +(Datenmenge-1)
     .               .                .              .
     .               .                .              .
     .               .                .              .
Quelladresse  +       0        -> Zieladresse +       0
```

Vor dem Ansprung der Routine muß in SOURCEL und SOURCEH die Quelladresse und in DRAINL und DRAINH die Zieladresse des zu verschiebenden Blocks festgelegt werden.

Im AMOUNTL und AMOUNTH wird die Anzahl der Daten in der 2-Byte- Adressform angegeben.

```
                SOURCEL  EQU $C000          QUELLADR. LOW
                SOURCEH  EQU $C001          QUELLADR. HIGH
                DRAINL   EQU $C002          ZIELADR. LOW
                DRAINH   EQU $C003          ZIELADR. HIGH
                AMOUNTL  EQU $C004          DATENMENGE LOW
                AMOUNTH  EQU $C005          DATENMENGE HIGH


                ************************************
                *                                  *
                *        BLOCKTRANSFER             *
                *                                  *
                ************************************

                         ORG $C010

C010: AD00C0             LDA SOURCEL
C013: 8562               STA 98             LADEN QUELLADRESSE NACH
C015: AD01C0             LDA SOURCEH        ZEROPAGE
C018: 8563               STA 99
C01A: AD02C0             LDA DRAINL
C01D: 8564               STA 100            LADEN ZIELADRESSE NACH
C01F: AD03C0             LDA DRAINH         ZEROPAGE
C022: 8565               STA 101
C024: AD04C0             LDA AMOUNTL
C027: 856A               STA 106            DATENMENGE LADEN
C029: AD05C0             LDA AMOUNTH
C02C: 856B               STA 107
C02E: A563               LDA 99
C030: C565               CMP 101
C032: B026               BCS BOTTOM         QUELLE > ZIEL!
C034: A563    TOP        LDA 99             QUELLE < ZIEL!
C036: 18                 CLC
C037: 656B               ADC 107            QUELLE:=QUELLE+MENGE HIGH
C039: A565               LDA 101
C03B: 18                 CLC
C03C: 656B               ADC 107            ZIEL:=ZIEL+MENGE HIGH
C03E: A46A               LDY 106            MENGE LOW ALS OFFSET
C040: B162    LOOP1      LDA (98),Y
C042: 9164               STA (100),Y        DATEN UMLADEN
C044: 88                 DEY
C045: C0FF               CPY #255           SEITENUEBERSCHREITUNG?
C047: D0F7               BNE LOOP1
C049: C663               DEC 99             QUELLE-256
C04B: C665               DEC 101            ZIEL-256
C04D: A56B               LDA 107
C04F: 38                 SEC
C050: E901               SBC #1
C052: 856B               STA 107            MENGE-256
C054: B001               BCS MARK1
C056: 60                 RTS                MENGE=0 D.H. FERTIG!
C057: 4C40C0  MARK1      JMP LOOP1
C05A: A000    BOTTOM     LDY #0             ANFANGSOFFSET=0
C05C: B162    LOOP2      LDA (98),Y
C05E: 9164               STA (100),Y        DATEN UMLADEN
C060: C46A               CPY 106            ZAEHLER=MENGE LOW?
C062: D005               BNE MARK2          NEIN!
C064: A56B               LDA 107            MENGE HIGH=0?
C066: D001               BNE MARK2          NEIN!
```

```
C068: 60               RTS          FERTIG!
C069: C8      MARK2    INY
C06A: D0F0             BNE LOOP2    KEINE SEITENUEBERSCHREITUNG ZAEHLER
C06C: E663             INC 99       QUELLE+256
C06E: E665             INC 101      ZIEL+256
C070: C66B             DEC 107      MENGE-256
C072: 4C5CC0 MARK3     JMP LOOP2
PHYSICAL ENDADDRESS: $C075

*** NO WARNINGS

SOURCEL          $C000              SOURCEH          $C001
DRAINL           $C002              DRAINH           $C003
AMOUNTL          $C004              AMOUNTH          $C005
TOP              $C034    UNUSED    LOOP1            $C040
MARK1            $C057              BOTTOM           $C05A
LOOP2            $C05C              MARK2            $C069
MARK3            $C072    UNUSED
```

NOTIZEN

Grafikbildschirm löschen

Wenn Sie das vorangegangene BASIC-Programm eingetippt und gestartet haben, dann haben Sie sicher festgestellt, daß das Löschen des Graphikbildschirms (Zeile 10) sehr lange Zeit benötigt.

Abhilfe schafft hier ebenfalls ein Maschinenprogramm, das diese Arbeit in einem Bruchteil der Zeit erledigt.

In Adresse 830 and 831 dec wird dem Programm mitgeteilt, ab welcher Adresse der Graphikbildschirm liegt.

Z. B. Anfangsadresse Graphikbildschirm = 8192

dann 830 = 0 831 = 32

Der Ansprung der Routine erfolgt auf Adresse 832.

```
                ADREND   EQU 829
                ADRLOW   EQU 830
                ADRHIG   EQU 831


                ************************************
                *                                  *
                * GRAPHICBILDSCHIRM LOESCHEN       *
                *                                  *
                ************************************


                         ORG 832,$B000

0340: AD3F03             LDA ADRHIG
0343: 85FC               STA 252
0345: 18                 CLC
0346: 6920               ADC #32
0348: 8D3D03             STA ADREND
034B: AD3E03             LDA ADRLOW
034E: 85FB               STA 251
0350: A200     MAR2      LDX #0
0352: 8A                 TXA
0353: 81FB               STA (251,X)
0355: E6FB               INC 251
0357: D002               BNE MAR1
0359: E6FC               INC 252
035B: A5FB     MAR1      LDA 251
035D: D0F1               BNE MAR2
035F: A5FC               LDA 252
0361: CD3D03             CMP ADREND
0364: D0EA               BNE MAR2
0366: 60                 RTS
PHYSICAL ENDADDRESS: $B027

*** NO WARNINGS

ADREND              $033D                ADRLOW              $033E
ADRHIG              $033F                MAR2                $0350
MAR1                $035B
```

6502 Opcodes

low / high	0	1	2	3	4	5	6	7	low / high
0	BRK 0	ORA (Ind,X) 1	 2	 3	 4	ORA 0-pg 5	ASL 0-pg 6	 7	0
1	BPL 16	ORA (Ind),Y 17	 18	 19	 20	ORA 0-pg,X 21	ASL 0-pg,X 22	 23	1
2	JSR 32	AND (Ind,X) 33	 34	 35	BIT 0-pg 36	AND 0-pg 37	ROL 0-pg 38	 39	2
3	BMI 48	AND (Ind),Y 49	 50	 51	 52	AND 0-pg,X 53	ROL 0-pg,X 54	 55	3
4	RTI 64	EOR (Ind,X) 65	 66	 67	 68	EOR 0-pg 69	LSR 0-pg 70	 71	4
5	BVC 80	EOR (Ind),Y 81	 82	 83	 84	EOR 0-pg,X 85	LSR 0-pg,X 86	 87	5
6	RTS 96	ADC (Ind,X) 97	 98	 99	 100	ADC 0-pg 101	ROR 0-pg 102	 103	6
7	BVS 112	ADC (Ind),Y 113	 114	 115	 116	ADC 0-pg,X 117	ROR 0-pg,X 118	 119	7
8	 128	STA (Ind,X) 129	 130	 131	STY 0-pg 132	STA 0-pg 133	STX 0-pg 134	 135	8
9	BCC 144	STA (Ind),Y 145	 146	 147	STY 0-pg,X 148	STA 0-pg,X 149	STX 0-pg,Y 150	 151	9
A	LDY Imm 160	LDA (Ind,X) 161	LDX Imm 162	 163	LDY 0-pg 164	LDA 0-pg 165	LDX 0-pg 166	 167	A
B	BCS 176	LDA (Ind),Y 177	 178	 179	LDY 0-pg,X 180	LDA 0-pg,X 181	LDX 0-pg,Y 182	 183	B
C	CPY Imm 192	CMP (Ind,X) 193	 194	 195	CPY 0-pg 196	CMP 0-pg 197	DEC 0-pg 198	 199	C
D	BNE 208	CMP (Ind),Y 209	 210	 211	 212	CMP 0-pg,X 213	DEC 0-pg,X 214	 215	D
E	CPX Imm 224	SBC (Ind,X) 225	 226	 227	CPX 0-pg 228	SBC 0-pg 229	INC 0-pg 230	 231	E
F	BEQ 240	SBC (Ind),Y 241	 242	 243	 244	SBC 0-pg,X 245	INC 0-pg,X 246	 247	F
high / low	0	1	2	3	4	5	6	7	high / low

low / high	8	9	A	B	C	D	E	F	low / high
0	PHP 8	ORA Imm 9	ASL A 10	11	12	ORA Abs 13	ASL Abs 14	15	0
1	CLC 24	ORA Abs,Y 25	26	27	28	ORA Abs,X 29	ASL Abs,X 30	31	1
2	PLP 40	AND Imm 41	ROL A 42	43	BIT Abs 44	AND Abs 45	ROL Abs 46	47	2
3	SEC 56	AND Abs,Y 57	58	59	60	AND Abs,X 61	ROL Abs,X 62	63	3
4	PHA 72	EOR Imm 73	LSR A 74	75	JMP Abs 76	EOR Abs 77	LSR Abs 78	79	4
5	CLI 88	EOR Abs,Y 89	90	91	92	EOR Abs,X 93	LSR Abs,X 94	95	5
6	PLA 104	ADC Imm 105	ROR A 106	107	JMP (Ind) 108	ADC Abs 109	ROR Abs 110	111	6
7	SEI 120	ADC Abs,Y 121	122	123	124	ADC Abs,X 125	ROR Abs,X 126	127	7
8	DEY 136	137	TXA 138	139	STY Abs 140	STA Abs 141	STX Abs 142	143	8
9	TYA 152	STA Abs,Y 153	TXS 154	155	156	STA Abs,X 157	158	159	9
A	TAY 168	LDA Imm 169	TAX 170	171	LDY Abs 172	LDA Abs 173	LDX Abs 174	175	A
B	CLV 184	LDA Abs,Y 185	TSX 186	187	LDY Abs,X 188	LDA Abs,X 189	LDX Abs,Y 190	191	B
C	INY 200	CMP Imm 201	DEX 202	203	CPY Abs 204	CMP Abs 205	DEC Abs 206	207	C
D	CLD 216	CMP Abs,Y 217	218	219	220	CMP Abs,X 221	DEC Abs,X 222	223	D
E	INX 232	SBC Imm 233	NOP 234	235	CPX Abs 236	SBC Abs 237	INC Abs 238	239	E
F	SED 248	SBC Abs,Y 249	250	251	252	SBC Abs,X 253	INC Abs,X 254	255	F
high / low	8	9	A	B	C	D	E	F	high / low

ASCII Character Codes

DECIMAL	CHAR.	DECIMAL	CHAR.	DECIMAL	CHAR.
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	\
007	BEL	050	2	093	]
008	BS	051	3	094	↑
009	HT	052	4	095	←
010	LF	053	5	096	`
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

LF=Line Feed FF=Form Feed CR=Carriage Return DEL=Rubout

ASCII Symbols

NUL—Null
SOH—Start of Heading
STX—Start of Text
ETX—End of Text
EOT—End of Transmission
ENQ—Enquiry
ACK—Acknowledge
BEL—Bell
BS—Backspace
HT—Horizontal Tabulation
LF—Line Feed
VT—Vertical Tabulation
FF—Form Feed
CR—Carriage Return
SO—Shift Out
SI—Shift In

DLE—Data Link Escape
DC—Device Control
NAK—Negative Acknowledge
SYN—Synchronous Idle
ETB—End of Transmission Block
CAN—Cancel
EM—End of Medium
SUB—Substitute
ESC—Escape
FS—File Separator
GS—Group Separator
RS—Record Separator
US—Unit Separator
SP—Space (blank)
DEL—Delete

Base Conversion Table

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	41952
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

To convert from hex to decimal, read across the top line for the first hex digit, then down to the second hex digit. For example, 6A hex is 166 in decimal (see shading).

To convert from decimal to hex, find the decimal number to be converted. Next, read up for the first hex digit, and over for the second hex digit.

NOTIZEN

Umwandlung von Ziffern — Strings in Flieskommazahlen

Ein häufig auftretendes Problem bei der Programmierung in Maschinensprache ist die Umwandlung von Ziffernfolgen, die als ASCII-Strings irgendwo im Speicher stehen, in eine Zahl, mit der arithmetische Operationen vorgenommen werden können.

Nehmen wir an, wir haben uns auf dem Bildschirm eine Eingabemaske erstellt, in der in der 5. Bildschirmzeile der Preis eines Produktes und in der 6. Zeile die Produktmenge angegeben werden kann. In der Zeile 7 soll nun der Gesamtpreis ausgegeben werden.

Eine Berechnung ist vorerst nicht möglich, da die Zahlen als Ziffernfolgen im Bildschirmspeicher stehen.

Mit Hilfe eines kleinen Maschinenprogramms können diese Strings jedoch in Zahlen umgewandelt werden, mit denen schließlich arithmetische Operationen möglich sind.

Für unser Beispiel ist die Vorgehensweise folgende:

Der Anfang des 1. Ziffernstrings wird in Zwei-Byte-Adressform festgelegt, z. B. Ziffernstring beginnt ab Adresse 2100 dez, d.h., Zwei-Byte-Adressform = 52,8.

Diese Adresse minus 1 wird der CHRGET-Routine (73-SA hex, 115-138 dez) mitgeteilt.

Das geschieht dadurch, daß man eine absolute Ladeadresse dieser Routine einfach mit der neuen Adresse überschreibt. Die Ladeadresse besteht aus den Zellen 7A und 7B hex = 122 und 123 dez. Die Anfangsadresse muß um den Wert 1 erniedrigt werden, da die CHRGET-Routine diese Adresse zu Beginn automatisch um 1 erhöht.

Eine zweite Routine, die ab Adresse BCF3 hex (= 48371 dez) zur Verfügung steht, und die sich ihrerseits der CHRGET-Routine bedient, wandelt den Ziffernstring in eine Fließkommazahl um, und legt sie im Fließkomma-akkumulator ab. (61-66 hex, 97 - 102 dez).

Bevor nun aber die Umwandlungsroutine angesprungen werden kann, muß die CHRGET-Routine durchlaufen werden, die entscheidet, ob eine Ziffernfolge vorliegt oder nicht. Beginnt der auszuwertende String nämlich mit einem Zeichen, das keiner Zahl entspricht (z.B. mit Buchstaben), dann wird als Wert die Zahl Null angenommen. Gültige erste Zeichen sind die Zahlen 0 bis 9, . , + , - , Leerzeichen (Space) werden von der CHRGET-Routine überlesen.

Die Umwandlung eines Ziffernstrings ist beendet, wenn die CHRGET-Routine auf ein Zeichen trifft, das keiner Zahl mehr entspricht (z.B. alle Buchstaben außer 'E'. das für Exponent steht).

Die fertig umgewandelte Zahl steht nun im Fließkomma-akkumulator zur Verfügung und kann für weitere Berechnungen verwendet werden.

ZUR BEACHTUNG:

Bei jedem Durchlauf der Umwandlungsroutine wird der zweite Fließkommaakku (ARG) überschrieben. Wollen Sie die erhaltene Zahl zur späteren Verwendung zwischenspeichern. dann sollten Sie sich einen Variablenbereich definieren (siehe Abschnitt: arithmetische Operationen in Maschinensprache).

Das nachfolgende Maschinenprogramm wandelt zwei Ziffernstrings (String 1 ab Adresse 832 und String 2 ab Adresse 852) in Zahlen um. addiert diese Zahlen und gibt sie schließlich in der 1. Bildschirmzeile aus.

Das kurze BASIC-Programm lädt die Ziffernstrings in den Speicher und startet das Maschinenprogramm. Experimentieren Sie. indem Sie die Daten in Zeile 40 und 50 verändern.

```
CHRGET   EQU $0073

         ORG $C000

*****************************************
*                                       *
* ADDITION ZWEIER ZAHLEN. DIE ALS       *
* ASCII ZEICHEN IM SPEICHER STEHEN      *
* (HIER: ZAHL1 AB 832, ZAHL2 AB 852     *
*                                       *
*****************************************

C000: A93F          LDA #63           ZAHL 1 AB ADRESSE 832 DEZ
C002: 857A          STA $7A
C004: A903          LDA #3
C006: 857B          STA $7B
C008: 207300        JSR CHRGET        1. ZEICHEN EINLESEN
C00B: 20F3BC        JSR $BCF3         UMWANDLUNG ASCII - FLIESSKOMMA
C00E: A278          LDX #120          ZAHL NACH 888 BIS 893 ABLEGEN
C010: A003          LDY #3
C012: 20D4BB        JSR $BBD4         FAC NACH KONSTANTE (888-893)
C015: A953          LDA #83           ZAHL 2 AB ADRESSE 852
C017: 857A          STA $7A
C019: 207300        JSR CHRGET        1. ZEICHEN HOLEN
C01C: 20F3BC        JSR $BCF3         UMWANDLUNG ASCII NACH FLIESSKOMMA
C01F: A978          LDA #120          ADRESSE 1. ZAHL
C021: A003          LDY #3
C023: 2067B8        JSR $B867         FAC=FAC+KONSTANTE
C026: 20DDBD        JSR $BDDD         FAC NACH ASCII
C029: A20A          LDX #10           AUSGABEPOSITION
C02B: A000          LDY #0            AUSGABEROUTINE
C02D: B90001 LOOP3  LDA $100,Y
C030: C900          CMP #0            ENDE STRING?
C032: D001          BNE KON1
C034: 60            RTS
C035: 9D0004 KON1   STA 1024,X        AUF BILDPOS. SETZEN
C038: A901          LDA #1
C03A: 9D00D8        STA 55296,X       FARBE SETZEN
C03D: C8            INY               SCHLEIFENZAEHLER ERHOEHEN
C03E: E8            INX
C03F: 4C2DC0        JMP LOOP3
PHYSICAL ENDADDRESS: $C042
```

```
10 FORI=0TO5:READA$:A=ASC(A$):POKE832+I,A:NEXT
20 FORI=0TO5:READA$:A=ASC(A$):POKE852+I,A:NEXT
30 SYS49152
40 DATA1,2,.,1,1,2
50 DATA2,0,2,.,1,0
```

Wichtige ROM-Routinen

Adresse hex	dez	Eingabe über	Ausgabe
E566	58726	-----	-------

Cursor home: Der Cursor wird auf die Home-Pos. (linke, obere Bildschirmecke) gesetzt.

E544	58692	Diese Routine löscht den Bildschirm und setzt den Cursor an die Home-Position.	
E8EA	59626	-----	-------

Bildschirm scrollen: Der Bildschirminhalt wird eine Zeile nach oben geschoben. Die oberste Zeile geht dabei verloren.

E5B4		-----	A

Zeichen aus Tastaturpuffer holen:

Ein Zeichen wird aus dem Tastaturpuffer geholt und sein ASCII-Code im Akku übergeben. Befindet sich im Tastaturpuffer kein Zeichen, dann beinhaltet der Akku eine Null.

E9158 A --------

Zeichenausgabe Bildschirm:

Das Zeichen, dessen ASCII-Code im Akku steht, wird an der aktuellen Cursorposition ausgegeben und der Cursor um eine Position nach rechts geschoben.
Steuerzeichen werden ebenfalls ausgeführt.

EA1C A,X --------

Zeichenausgabe mit Farbe auf Bildschirm:

Im Akku muß der BILDSCHIRMCODE des auszugebenden Zeichens und im X-Register seine Farbe (0-15) stehen. Die Ausgabe erfolgt an der aktuellen Cursorposition.

F157 ----- --------

BASIN:

Ein Zeichen wird vom Gerät geholt, das in Zelle $99 festgelegt wurde (default = 0 = Tastatur) und im Akku übergeben.

F1CA A --------

BSOUT:

Das Zeichen, dessen ASCII-Code im Akku steht, wird an das in Zelle $ 9A festgelegte Ausgabegerät übergeben (default = 3 = Bildschirm).

0073 ----- --------

CHRGET: Diese Routine holt das nächste Zeichen aus dem BASIC-Text.
Die Anfangsadresse der zu holenden Zeichen wird in Zelle $7A und $7B festgelegt. Die Adresse wird automatisch erhöht.
Der ASCII-Code des Zeichens wird im Akku übergeben.
Zwischenräume werden überlesen.
Handelt es sich bei dem eingelesenen Zeichen um keine Ziffer, dann ist die Carry-Flag gesetzt.

BCF3 ----- --------

ASCII-> Fließkomma: Diese Routine wandelt eine Zahl, bestehend aus ASCII-Zeichen, in Fließkommaformat um und legt sie im Fließkommaakkumulator ($62-$65) ab. Die Anfangsadresse der Zahl muß in Zelle $7A und $7B festgelegt werden.
Beginnt der String mit nicht-numerischen Zeichen, dann wird im Fließkommaakku eine Null übergeben.

E097 ----- --------

RND Die BASIC-Funktion RND legt nach Ansprung eine neue Zufallszahl in den Zellen $8B bis $8F ab.

EA87 60039 ----- --------

Tastaturabfrage: Die Tastatur wird abgefragt und der Tastaturpuffer wird aktualisiert.

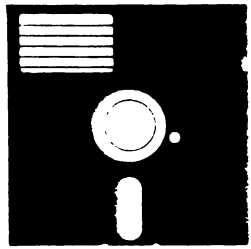
E50A	58634	X,Y,Carry=0	--------

Cursor setzen: Die Routine setzt den Cursor bei gelöschter Carry-Flag auf die angegebene Position. Im X-Register wird die Zeilenposition (0 - 24) und im Y-Register die Spaltenposition (0 - 39) mitgeteilt.

E50A	58634	Carry=1	X,Y

Cursor holen: Bei gesetzter Carry-Flag wird die aktuelle Cursorposition im X-Register (Spalte 0 - 39) übergeben.

Alle Programme aus diesem Buche auf Diskette



Verwenden Sie dieses Blatt als Bestellschein.

Liefern Sie an folgende Adresse:

Name . . . . . Vorname . . . . .

Straße . . . . . .

PLZ . . . . Ort . . . . . .

Alle Programme aus dem Buch Nr. 189 zu DM 49,00 incl. Porto und Verpackung auf Diskette.

☐ BLIZTEXT – Spitzenwortprozessor zu DM 199,– auf Diskette.

☐ . . . . . .

☐ Ich werde den Betrag heut auf Ihr Postscheck-Konto München 15 994–807 überweisen.

☐ Bitte liefern Sie per Nachnahme. Hier kommen noch 6,50 DM Nachnahmegebühren hinzu.

. . . . . .

Datum Unterschrift (f. Jugendliche unter 18 Jahre der Erziehungsberechtigte)

NOTIZEN

Anhang

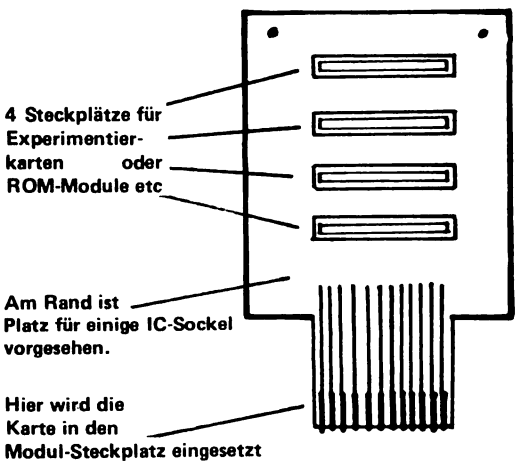
Rund um Ihren Commodore-64

Für denjenigen, der gerne seinen Commodore 64 erweitern möchte oder Peripherie selbst anschließen will, hat die Fa. Ing. W. Hofacker GmbH ein umfangreiches Stecker- und Platinenprogramm bereitgestellt.
Anhand der nachfolgenden Skizze können Sie die entsprechenden Stecker identifizieren und deren Bestell-Nr. und Preis ermitteln. Alle Stecker haben wir für Sie auf Lager (Zwischenverkauf vorbehalten).

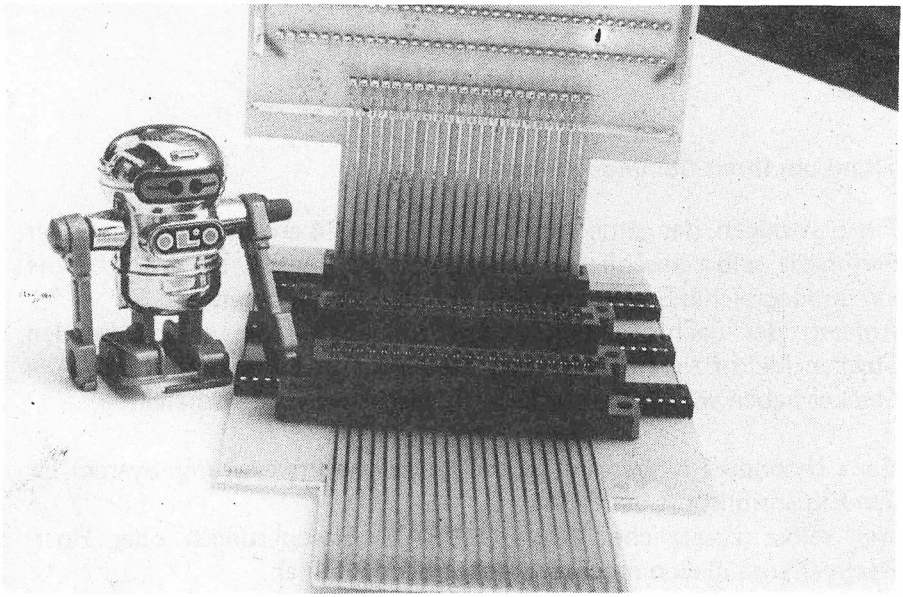
Ganz besonders interessant ist das preiswerte Erweiterungs-System für den Expansionsport des C-64.
Wer selbst zusätzliche I/O, ROM/RAM Erweiterungen oder Host-Rechner anschließen möchte, braucht diese Platinen.

Das System besteht aus zwei Platinen:

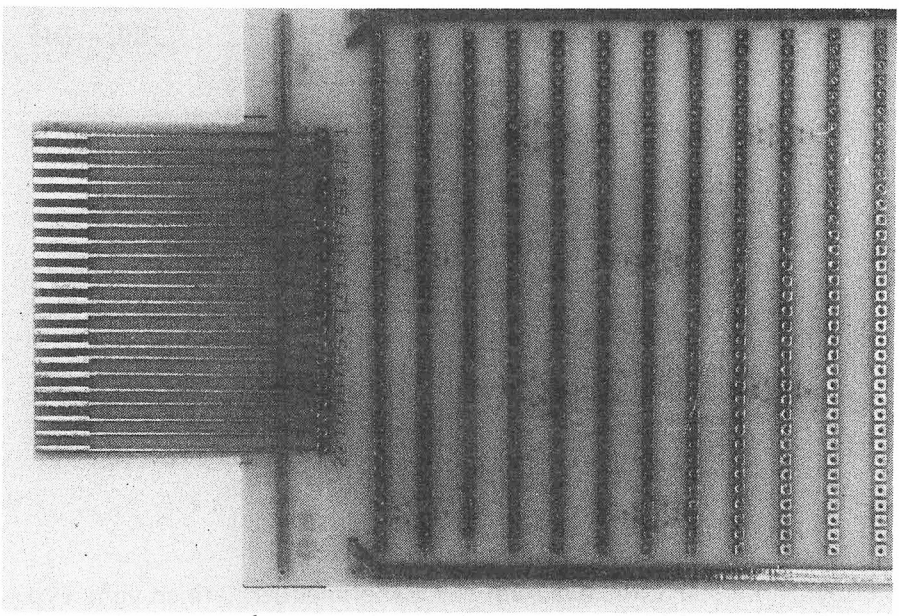
Best.-Nr. 4992 Expanisonsplatine (Bausatz) **99, – DM**



Sie dient in erster Linie dazu, um vier Experimentierplatinen vom Typ No. 4970 oder ROM Moduln oder andere Erweiterungsbausteine gleichzeitig an den Expansionsanschluß des C-64 zu stecken. Neben den Steckplätzen ist noch ein Experimentierfeld für einige IC's vorgesehen.



2 Stecker sind im Preis inbegriffen.
Best.-Nr. 4970 Experimentierplatine **39, – DM**



Diese Platine erlaubt Ihnen den Aufbau von eigenen Systemerweiterungen wie z. B. weitere I/O-Bausteine, zusätzliche Schnittstellen, RAM oder ROM-Erweiterungen u. v. a. mehr.

Die Platine ist durchkontaktiert, Masseleitung und +5V sind herausgeführt. Im einfachsten Falle läßt sich die Platine auch als Stecker verwenden.

44poliger Stecker
Best.-Nr. 4996/A
DM 39,– / Stück

ERWEITERUNGSPORT

DIN Stecker
5polig
Monitor

DIN Stecker
6polig
RS232

Cassetten-
anschlußstecker
Best.-Nr. 4996
DM 9,80

USER PORT
Stecker
Best.-Nr. 4847
DM 19,80

Joystickport 9polig
Best.-Nr. 7040

DM 9,80 / Stück

Best.-Nr. 4992 Expansionplatine DM 99,–

Best.-Nr. 4970 DM 39,–
Externe Experimentierplatine

Ing. W. Hofacker GmbH
Tegernseer Str. 18
D-8150 Holzkirchen

Tel.: 08024/ 7331
Tlx.: 52 69 73

MACROFIRE

Der Macro-Assembler für den Commodore-64 von HOFACKER

Ein extrem leistungsfähiges Werkzeug für den Programmentwickler. Sehr hohe Geschwindigkeit. Übersetzt ca. 10K Quelltext in 3 – 4 Sekunden.

Dieser Assembler wurde weltweit bereits mehrere hundert Mal verkauft und wird bei führenden US-Softwarefirmen in der System- und Anwendersoftwareentwicklung eingesetzt.

Das Paket besteht aus einem Editor (ähnlich unserem Wortprozessor BLIZTEXT), dem Assembler und einem komfortablen Maschinensprachen-Monitor.
Das Paket ist integriert. Sie erreichen den Assembler oder den Monitor vom Editor aus über einen Tastendruck. Es wird in drei Durchgängen übersetzt. Bis zu 1000 Labels können vereinbart werden. Volle Include-Funktion auf Cassette oder Diskette. Sehr komfortable Anwenderführung, ca. 45 Kommandos im Editor. Alle Commodore-Disketten und Cassetten I/O-Befehle werden unterstützt. Unseren Kenntnissen nach einer der leistungsfähigsten Editor/Assembler-Pakete weltweit.

Lieferung erfolgt vorerst mit englischem Manual. Ein deutsches Handbuch wird nachgereicht. Eine einmalige, an die Person gebundene Benutzerlizenz kostet 199,– incl. MwSt.

Best.-Nr. 4964 **199,– DM**

BLIZTEXT TM

Nach unserem Wissen einer der besten Wortprozessoren für den C-64

Ab sofort liefert die Firma Ing. W. Hofacker GmbH in Holzkirchen ein sehr leistungsfähiges Textverarbeitungsprogramm für den Commodore-64.
Dieses neue Programm erlaubt es dem Anwender, ein Textsystem zu einem noch bis heute undenkbar niedrigen Preis zusammenzustellen.

Erforderliche Hardware: (Minimum)

1 Commodore-64
1 Cassettenrecorder oder Diskette 1541
1 Drucker, ähnlich VC1525
1 Bildschirmmonitor oder Fernsehgerät

Das Textverarbeitungsprogramm besteht aus drei Teilen, dem Editor, dem Formatter und einem Terminalprogramm.
Alle drei Systemelemente sind zu einem ergonomischen und leicht zu bedienenden Softwarepaket integriert.
Das Programm bietet dem Anwender praktisch alles, was er von einer professionellen Textverarbeitung erwarten kann.
Hier kurz eine Zusammenfassung der wichtigsten Eigenschaften:

1.) Voll bildschirmorientiert, horizontales und vertikales Scrolling bis zu 255 Zeichen / Zeile
2.) Statusanzeige für Zeile und Zeichen in der Zeile
3.) 27KRAM frei für Text im Hauptspeicher plus 4K Kopierpuffer
4.) Echte Include-Funktion auf Cassette und Diskette. Dies bedeutet, daß Sie Textfiles auf Diskette in den momentanen Text einbinden können.
5.) Linker und rechter Randausgleich sowie Zentrierung sind natürlich selbstverständlich
6.) Ca. 30 Kommandos stehen im Editor zur Verfügung, weitere 20 Befehle zur Textformattierung. Alle Commodore-Cassetten und Disk-I/O-Befehle werden voll unterstützt.
7.) Verbesserte Fehlerkanal-Leseoperation, Directory-Abfrage durch 2 Tastenbetätigungen
8.) Groß- und Kleinschreibung auf praktisch alle Drucker (RS232, IEEE, IEC)
9.) Dynamische Formattierung durch direkt in den Text eingefügte Kommandos
10.) Komfortable Fehlermeldungen
11.) Druckersteuerzeichen können individuell in den Text eingebaut werden. Somit können Sie auf fast allen Druckern unterstreichen, Breitschrift, Fettschrift usw.
12.) **Eingebauter Terminalmodus** – Dies hat es bis jetzt weltweit noch nicht gegeben. Ein Terminal kann zum Senden und Empfangen von elektronischer Post simuliert werden. Der Modus kann auch zur Verwendung von Rechnerkopplungen mit anderen Personalcomputern verwendet werden.

Ein Produkt, für das wir Ihnen praktisch eine Zufriedenheitsgarantie geben könnten.

Ein Manual mit ca. 70 Seiten in englischer Sprache ist verfügbar. Ein ausführliches, deutsches Handbuch ist in Arbeit. Verfügbar auf Cassette oder Diskette.
Der Wortprozessor "BLIZTEXT" kann sofort bestellt werden. Der Preis beträgt **DM 199,00** incl. Mwst. und englischem Handbuch. Ein deutsches Handbuch wird nachgeliefert.
Das im Preis enthaltene Handbuch wurde mit BLIZTEXT selbst erstellt und kann für **DM 49,00** alleine bezogen werden. Der Kaufpreis wird später angerechnet.

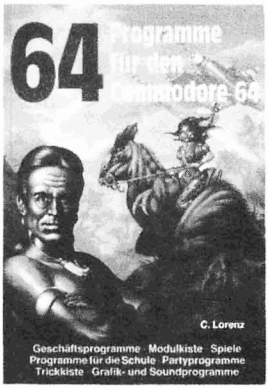
Ing. W. Hofacker GmbH, Tegernseer Str. 18, D-8150 Holzkirchen, Tel.: 08024 / 73 31, Tlx.: 52 69 73

Bücher von HOFACKER

Beherrschen Sie Ihren C-64

Auf über 120 Seiten findet der C-64 Besitzer die Zusatzinformationen, die er noch zusätzlich zum mitgelieferten Handbuch benötigt. Grundlagen, ein Blick ins Innere von BASIC, einfacher Programmschutz, Listschutz-Tricks und Programmiertips. Einführung in Dateien auf Cassette und Diskette. Ton und Grafik, nützliche Hilfsprogramme und hochauflösende Grafik.

Best.-Nr. 147 **19,80 DM**



64 Programme für den C-64

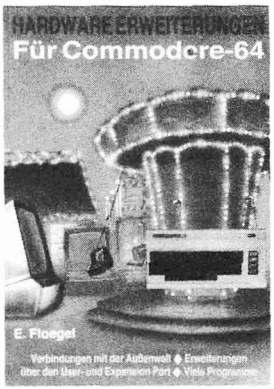
Dieses Buch braucht jeder C-64 Besitzer. Sie finden darin Geschäftsprogramme für den C-64, nützliche BASIC-Programme (Modulkiste), Progr. für den Schüler u. Studenten (Mathematik und Statistik, Spiele, Partyprogramme, verschiedene Programme mit Grafik und Sound, sowie Tips und Tricks. 210 Seiten.

Best.-Nr. 145 **39, – DM**

BASIC Programmier-Handb.

Dieses Buch braucht jeder C-64-, Atari-, Sinclair- und APPLE-Besitzer. Es eignet sich für alle heute am Markt befindlichen Personalcomputer. Dem Leser werden die Grundelemente von BASIC mitgeteilt, auch soll es helfen, selbst Programme zu schreiben. Die Anfänger finden einen BASIC-Einsteiger Kurs. Dem erfahrenen Programmierer dient es als Nachschlagewerk. 154 S.

Best.-Nr. 113 **19,80 DM**



Hardware Erweiterungen für den Commodore-64

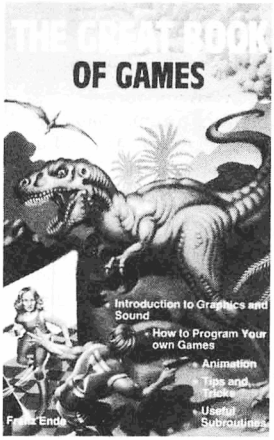
Ein ausführliches Anleitungsbuch für jeden der seinen C-64 für Steuerungen und Hardware Experimente verwenden will. Verbindungen mit der Außenwelt über den User- u. Expansion Port, Erweiterungen, viele Programme. 160 Seiten.

Best.-Nr. 146 **39,00 DM**

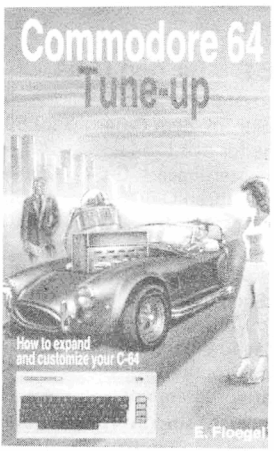
Programmieren in Maschinensprache mit dem C-64

Einführung in die 6502/10 Maschinensprache auf dem C-64. Sehr viele Beispiele. Der gesamte Befehlssatz. Ein ausgezeichnetes Buch. 200 Seiten.

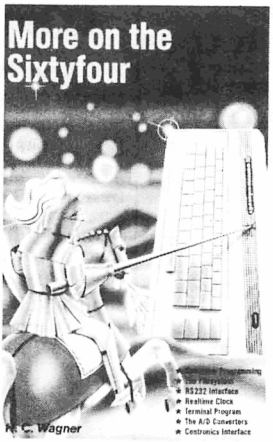
Best.-Nr. 124 **29,80 DM**



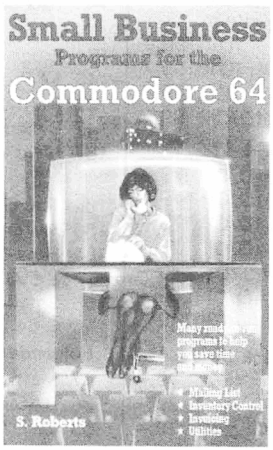
The Great Book of Games
Vol. I – 46 Games f. the C-64
Wie programmiere ich eigene Spiele. Einführung in Graphik, Tonausgabe und Bewegung mit sehr vielen Beispielen. 144 Seiten (englisch).
Best.-Nr. 182 **29,80 DM**



Commodore 64 - Tune up
Ein ausführliches Anleitungsbuch für jeden, der seinen C-64 für Steuerungen und Hardware Experimente verwenden will. Übersetzt von Nr. 146 (englisch).
Best.-Nr. 185 **39 00 DM**



More on the Sixtyfour (64)
Eine echte Sensation weltweit! Ein Buch voll gestopft mit wichtigen und nützlichen Maschinenprogr., Ein-/Ausgabe von Text, rekursive Routinen, Echtzeituhr, RS-232 u. Centronics Treiber, A/D-Wandler am Joystickport u. v. m. 147 Seiten (englisch).
Best.-Nr. 183 **39,00 DM**



Small Business Programs for the Commodore-64
Geschäftsprogramme für den Commodore 64. 120 S. (engl.)
Best.-Nr. 186 **49,00 DM**

Weitere Bücher folgen !

How to Program your C-64
in 6502/10 Machine Language
Einführung in die 6502/10 Maschinensprache auf dem C-64. Übersetzt von Nr. 124. 108 Seiten (englisch).
Best.-Nr. 184 **29,80 DM**

Mehr als 29 Programme für den Commodore 64
Eine sehr interessante Programmsammlung für den C-64. 30 wertvolle Programme wie Sprite-Editor, Terminkalender, Schlange, Kalender, Statistik in BASIC, 3-D-Plot, Landesimulation, Konvertierung in andere Zahlensysteme, Ballspiel, u. a. (161 Seiten).
Best.-Nr. 187 **DM 29,80**

Neuheiten — Die SUPERSENSATION

BLIZMERGE für BLIZTEXT ★

Diese Adressenverwaltung erlaubt es, Ihnen Massenbriefe (aus Bliztext) mit verschiedenen Adressen automatisch zu schreiben.

Best.-Nr. 4945 NEU **99,– DM**

BLIZTEXT 1.1

Der Superwortprocessor für C-64. Voll bildschirmorientiert. Mehr als 70 Kommandos. 72 Seiten dt. Anleitung. Terminal-Software f. Netzwerke ist enthalten.

Der neue Bliztext erlaubt jetzt zusätzlich: Mailmerge, Kompletter Terminal Modus, Editierung v. BASIC-Programmen, Angebote schreiben mit Rechenoperationen.

Best.-Nr. 4965 **199,– DM**

(Aufpreis für Bliztextbesitzer DM 10,– plus Rückporto)

BUSIPACK I NEU

Das ist ein echter Durchbruch ! Lagerverwaltung mit Mindestmengen und Adressenverwaltung mit Fakturierung. Rechnungen schreiben, Lager wird mitgeführt, Adressen aus der Verwaltung. Ideal für jeden Kleinbetrieb.

Best.-Nr. 4963 **299,– DM**

Handbuch vorab (wird angerechnet) **49,– DM**

4962	Super Mailinglist: Adressenverwaltung bis 2000 Adressen pro Diskette m. C-64, sehr leistungsfähig	(D)	199,–
4961	Superinventory	(D)	199,–
4980	Adresskartei – 64	(C)	49,–
4954	Fakturierung m. Text	(D/C)	99,–

SPIELE für den C-64

4950	Spielepaket I	(D/C)	79,–
4951	Spielepaket II	(D/C)	79,–
4956	Mathematikprogramme	(D/C)	79,–
4986	Astrologie für C-64	(D/C)	79,–
4940	Shaft Raider-64 NEU	(D)	49,–
4941	GNOM	(D/C)	49,–
4942	Raingame	(D/C)	49,–

Buch/Disketten Pakete im SB-Pack für C-64

4700	Games for the C-64	**79,–**
4701	More on the 64	**79,–**
4702	How to program machine language	**79,–**

COMMODORE 64

Neue Hardware Erweiterungen

EPSON Printer KIT für Commodore-64

Software und Anleitung zum Bau einer Schnittstelle zum Anschluß von EPSON und STAR Drucker (Disk oder Cassette).

Best.-Nr. 4990 **59,– DM**

Super Sprite Editor für C-64 Neu

Zeichnen von Sprites mit dem Joystick, Mehrfarbenmodus, vergrössern u. verkleinern, simultane Darstellung, Spiegelung über die X- und Y-Achse, punktsymmetrisches Spiegeln, Ausgabe auf einen Drucker mit oder ohne Gitter, Ausgabe in verschiedenen Farben auf dem Drucker (als Schattierung).

Best.-Nr. 4946 **49,– DM**

BLIZTEXT Anwendungsbericht I über Datenübertragung. NEU

Best.-Nr. 4947 **19,80 DM**

4970	Externe Experimentierplatine am Modul Steckplatz	39,–
4992	Expansionsb. f. Modul Steckpl. (Bausatz). Erlaubt bis zu 4 Best.-Nr. 4970.	99,–
4847	User Port Stecker 24pol.	19,80
7040	Joystickportstecker (Weibchen)	9,80
4996	Cassettenportstecker	9,80

★ Bücher für den C-64

In deutscher Sprache

124	Progr. i. Ma.-Spr. m. dem C-64	29,80
145	64 Programme für den C-64	39,–
146	Hardware Erweiterungen f. C-64	39,–
147	Beherrschen Sie Ihren C-64	19,80
187	Mehr als 29 Progr. f. C-64 NEU	29,80

In englischer Sprache

182	The Great Book of Games, Vol. I	29,80
183	More on the Sixtyfour	39,–
184	How to Progr. your C-64 in i. 6502/10 Machine Language	29,80
186	Small Business Programs for the C-64 (Geschäftsprogramme)	49,–

Folgende Bücher (engl.) sind in Vorbereitung:

47	Mathematics, Statistics in BASIC	19,80
36	BASIC in 60 Minutes – a day	29,80
55	29 Programs for the C-64	29,80

Maschinensprache Utilities – C-64

MACROFIRE – Macroassembler für C-64 Editor/Assembler voll bildschirmorientiert. Include von Disk oder Cassette sehr schnell.

4964 (D/C) 199,–

4960	FORTH für C-64	(D)	299,–
4983	Miniassembler für C-64	(C)	49,–
4984	Maschinensprachemonitor	(C)	39,80
4985	Disassembler	(C)	29,80
4987	SUPERMON – 64	(D/C)	39,80

NOTIZEN

NOTIZEN

NOTIZEN

NOTIZEN

Weitere interessante Bücher von Hofacker:

Best.-Nr.	Titel	Preis / DM
	BÜCHER in deutscher Sprache	
1	Transistor Berechnungs- und Bauanleitungsbuch-1	29,80
2	Transistor Berechnungs- und Bauanleitungsbuch-2	19,80
3	Elektronik im Auto	9,80
4	IC-Handbuch, TTL, CMOS, Linear	19,80
5	IC-Datenbuch, TTL, CMOS, Linear	9,80
6	IC-Schaltungen, TTL, CMOS, Linear	19,80
7	Elektronik Schaltungen	19,80
8	IC-Bauanleitungs-Handbuch	19,80
9	Feldeffekttransistoren	9,80
10	Elektronik und Radio	19,80
12	Beispiele integrierter Schaltungen (BIS)	19,80
13	HEH, Hobby Elektronik Handbuch	9,80
16	CMOS Teil 1, Einführung, Entwurf, Schaltbeispiele	19,80
17	CMOS Teil 2, Entwurf und Schaltbeispiele	19,80
18	CMOS Teil 3, Entwurf und Schaltbeispiele	19,80
19	IC-Experimentier Handbuch	19,80
20	Operationsverstärker	19,80
21	Digitaltechnik Grundkurs	19,80
22	Mikroprozessoren, Eigenschaften und Aufbau	19,80
23	Elektronik Grundkurs, Kurzlehrgang Elektronik	9,80
24	Programmieren in Maschinensprache mit Z80, II	29,80
25	68000 Microcomputer Einführung	39,00
26	Mikroprozessor, Teil 2	19,80
27	BASIC-M für 6800/09/68000 (Motorola)	29,80
28	Lexikon u. Wörterbuch f. Elektr. u. Mikroprozessor	29,80
30	Floppy Disk Selbstbau-Handbuch (i. V.)	49,00
31	57 Praktische Programme in BASIC	39,00
32	ATARI BASIC, für Selbststudium und Praxis	39,00
33	Microcomputer Programmierbeispiele	19,80
34	TINY-BASIC Handbuch	19,80
35	Der freundliche Computer	29,80
102	Mathematische + Wissenschaftliche Progr. i. BASIC	29,80
103	Oszillographen-Handbuch	19,80
104	Portable Computer Handbuch (i. V.)	39,00
108	Rund um den Spectrum (Progr., Tips und Tricks)	29,80
109	6502 Microcomputer Programmierung	29,80
110	Programmierhandbuch für PET	29,80
111	Programmieren mit TRS-80 (GENIE)	29,80
112	PASCAL-Programmier-Handbuch	29,80
113	BASIC-Programmier-Handbuch (mit BASIC-Kurs)	19,80
114	Der Microcomputer im Kleinbetrieb	39,80
115	Dragon 6809 Handbuch (i. V.)	49,00
116	Einführung 16-Bit Microcomputer	29,80
118	Programmieren in Maschinensprache mit dem 6502	49,00
119	Programmieren in Maschinensprache mit Z80, I.	39,00
120	Anwenderprogramme für TRS-80 und GENIE	29,80
121	Microsoft BASIC-Handbuch	29,80
122	BASIC für Fortgeschrittene	39,00
123	IEC-Bus Handbuch	19,80
124	Progr. in Maschinensprache mit Commodore-64	29,80
127	Einführung i. d. Microcomputer-Progr. mit 6800	49,00
128	Programmieren mit dem CBM	29,80
130	Programmierbeispiele für CBM	9,80
132	CP-M Handbuch	19,80
133	Handbuch für MS-DOS	29,80
136	Funktionsanalyse	79,00
137	FORTH – Grundlagen, Einführung, Beispiele	49,00
139	BASIC für blutige Laien (speziell f. TRS-80, Genie)	19,80
140	Progr. i. BASIC u. Maschinencode mit dem ZX81	29,80
141	Progr. f. VC-20 (Spiele, Utilities, Erweiterungen)	29,80
143	35 Programme für den ZX81	29,80
144	33 Programme für den ZX-Spectrum	29,80
145	64 Programme für den Commodore-64	39,00
146	Hardware Erweiterungen für den Commodore-64	39,00
147	Beherrschen Sie Ihren Commodore-64	19,80
148	Programmierhandbuch für SHARP	49,00
149	Programme für TI 99/4A	49,00
175	Astrologie auf dem ATARI 800	49,00
187	Mehr als 29 Programme für den Commodore-64	29,80
188	Statistik in BASIC	39,00
189	6502 Maschinensprache - Beispiele für C-64	19,80
190	Das große Spielebuch für ATARI 600XL/800XL, I	29,80
200	FORTH-Anwendungsbeispiele	49,00
202	UNIX – Grundlagen und Anwendungen	39,00
204	Grafik und Ton mit dem Commodore-64	29,80
205	Das große Spielebuch für ATARI 600XL/800XL, II	29,80
212	Geschäftsprogramme für Commodore-64	39,00
213	Technische Gleichungssysteme in BASIC	49,00
217	Künstliche Intelligenz	19,80
8029	Z-80 Assembler-Handbuch	29,80
	BÜCHER in englischer Sprache	
29	Hardware Handbook	49,00
52	Small Business Programs for the IBM PC	29,80
151	8K Microsoft BASIC Reference Manual	9,80
152	Expansion Handbook for 6502 and 6800	19,80
156	Small Business Programs	29,80
158	The Second Book of Ohio Scientific	19,80
159	The Third Book of Ohio Scientific	29,80
160	The Fourth Book of Ohio Scientific	29,80
161	The Fifth Book of Ohio Scientific	19,80
162	ATARI Games in BASIC	19,80
164	ATARI-BASIC Learning by Using	19,80
166	Programming in 6502 Machinelanguage PET/CMB	49,00
169	How the Progr. y. ATARI i. 6502 Machinelanguage	29,80
170	FORTH on the ATARI – Learning by Using	29,80
171	See the Future with your ATARI (Astrology)	49,00
172	Hackerbook I (Tricks + Tips for your ATARI)	29,80
173	PD-Program Descriptions (ATARI)	9,80
174	ZX-81/TIMEX Progr. i. BASIC a. Machine Lang.	4,90
176	Programs + Tricks for VIC's	29,80
177	CP/M – MBASIC and the OSBORNE	29,80
178	The APPLE in Your Hand	39,00
182	The Great Book of Games Vol. I - Games f. the C-64	29,80
183	More on the Sixtyfour (Commodore-64)	39,00
184	How to Progr. your C-64 i. 6502/10 Machinelang.	29,80
185	Commodore-64 Tune-up	39,00
186	Small Business Programs for the Commodore-64	29,80

Der HOFACKER-Verlag produziert und vertreibt neben einer sehr großen Auswahl an Fachbüchern für Elektronik und Microcomputertechnik noch:

– Leerplatine und Bauanleitungen für Zusatzeinrichtungen für Ihren Personalcomputer, sowie

– Programme (Software) und Leercassetten (C-10) für die bedeutenden Personalcomputer.

(i. V. bedeutet: Buch ist in Vorbereitung)

HOFACKER

HOLZKIRCHEN **SINGAPORE** **LOS ANGELES**

ISBN 3-88963-189-4