# ATARI®

ATARI PROGRAMM

RXG 7075
1 Steckmodul/1 Diskette

ATARI MICROSOFT BASIC II

Program Copyright © 1982 by
Microsoft - All rights reserved

Für die deutsche Übersetzung

© 1983 Jegliche Rechte vorbehalten
ATARI ELEKTRONIK - Vertriebsges. mbH

ATARINSIDE

ATARI PROGRAMM

RXG 7075
1 Steckmodul/1 Diskette

# ATARI MICROSOFT BASIC II

Program Copyright © 1982 by
Microsoft - All rights reserved

Für die deutsche Übersetzung

© 1983 Jegliche Rechte vorbehalten
ATARI ELEKTRONIK - Vertriebsges. mbH

ATARINSIDE

# ATARI MICROSOFT BASIC BEFEHLE


| | | | | | |
|---|---|---|---|---|---|
| ABS | 47 | INKEY$ | 49 | RANDOMIZE | 42 |
| AFTER | 26 | INPUT | 31 | READ | 43 |
| ASC | 48 | INPUT..AT | 31 | REM | 43 |
| ATN | 47 | INSTR | 49 | RENUM | 23 |
| AUTO | 18 | INT | 47 | RESTORE | 44 |
| CHR$ | 48 | KILL | 20 | RESUME | 44 |
| CLEAR | 26 | LEFT$ | 49 | RETURN | 45 |
| CLEAR STACK | 26 | LEN | 49 | RIGHT$ | 49 |
| CLOAD | 18 | LET | 31 | RND | 47 |
| CLOSE | 27 | LINE INPUT | 32 | RUN | 24 |
| CLS | 60 | LINE INPUT..AT | 32 | SAVE | 24 |
| COLOR | 57 | LIST | 20 | SAVE...LOCK | 24 |
| COMMON | 27 | LOAD | 21 | SCRN$ | 50 |
| CONT | 19 | LOCK | 22 | SETCOLOR | 59 |
| COS | 47 | LOG | 47 | SGN | 48 |
| CSAVE | 19 | MERGE | 22 | SIN | 48 |
| DATA | 43 | MID$ | 49 | SOUND | 64 |
| DEF | 27 | MOVE | 32 | SQR | 48 |
| DEFSGN | 12 | NAME...TO | 22 | STACK | 45 |
| DEFDBL | 13 | NEW | 23 | STATUS | 52 |
| DEFINT | 12 | NEXT | 32 | STOP | 45 |
| DEFSTR | 14 | NOTE | 33 | STR$ | 50 |
| DEL | 19 | ON ERROR | 33 | STRING$(n,A$) | 50 |
| DIM | 28 | ON...GOSUB | 34 | STRING$(n,M) | 50 |
| DOS | 20 | ON...GOTO | 34 | TAN | 48 |
| END | 29 | OPEN | 35 | TIME | 53 |
| EOF | 51 | OPTION BASE | 36 | TIME$ | 51 |
| ERL | 51 | OPTION CHR | 36 | TROFF | 25 |
| ERR | 51 | OPTION PLM | 37 | TRON | 25 |
| ERROR | 29 | OPTIN RESERVE | 37 | UNLOCK | 25 |
| EXP | 47 | PEEK | 52 | USR | 54 |
| FILL | 60 | PLOT | 59 | VAL | 51 |
| FOR..TO..STEP | 29 | PLOT...TO | 59 | VARPTR | 45 |
| FRE(0) | 52 | POKE | 52 | VERIFY | 25 |
| GET | 29 | PRINT | 37 | WAIT...AND | 46 |
| GOSUB | 30 | PRINT...AT | 38 | + | 48 |
| GOTO | 30 | PRINT...SPC | 38 | !(Anmerkung) | 43 |
| GRAPHICS | 56 | PRINT...TAB | 38 | '(Anmerkung) | 43 |
| IF...THEN | 30 | PRINT USING | 39 | | |
| IF..THEN..ELSE | 31 | PUT | 42 | | |


ATARINSIDE

# INHALT


VORWORT III

1. Neue Möglichkeiten und Wege der Programmierung 1
   Mindestspeicherplatz Ihres Computer Systems 2
   Einladen des Microsoft BASIC II 2
   Kontrollangaben auf dem Bildschirm 3
   Kopieren der Microsoft BASIC II Erweiterungsdiskette 3
   Beginn der Programmierung 5
     direkte Eingaben / Eingaben durch eine Programmzeile 5
     Programmierworte (Keywords) 5
     die Microsoft BASIC II Befehlszeile 5
     Bedeutung der Interpunktion beim Microsoft BASIC II 5
     Verbessern (Editieren) bestehender Eingaben 7
     Tasten mit Sonderfunktionen 9

2. Bestandteile eines Programms 11
   Konstante und Variable 11
     Korrekte Variablennamen 11
     Genauigkeit numerischer Variablen 11
     ganzzahlige Konstante (Integer) 11
     ganzzahlige Variable 11
     Bestimmen einer Ganzzahl (Befehl DEFINT) 12
     Konstante mit einfacher Genauigkeit (single) 12
     Variable mit einfacher Genauigkeit 12
     Bestimmen der einfachen Genauigkeit (Befehl DEFSNG) 12
     Konstante mir doppelter Genauigkeit (double) 13
     Variable mit doppelter Genauigkeit 13
     Bestimmen der doppelten Genauigkeit (Befehl DEFDBL) 13
     Hexadezimale Konstante 13
   Alphanumerische (STRING-)($) Eingaben / Matrix 13
     Alphanumerische Konstanten 14
     Alphanumerische Variablen 14
     Bestimmen eines STRING (Befehl DEFSTR) 14
     Benutzen einer Matrix (ARRAY) 15
   Arithmetische, vergleichende und logische Operationen 16
     arithmetische Berechnungen (Operationen) 16
     Vergleiche 16
     logische Berechnungen 17

3. Programm Befehle (Kommandos) 18
   AUTO 18
   CLOAD 18


ATARINSIDE

CONT 18
CSAVE 19
DEL 19
DOS 20
KILL 20
LIST 20
LOAD 21
LOCK 22
NAME...TO 22
NEW 23
RENUM 23
RUN 24
SAVE 24
SAVE...LOCK 24
TROFF 25
TRON 25
UNLOCK 25
VERIFY 25

4. PROGRAMM ANWEISUNGEN (STATEMENTS) 26
AFTER 26
CLEAR 26
CLEAR STACK 26
CLOSE 27
COMMON 27
DEF 27
DIM 28
END 29
ERROR 29
FOR...TO...STEP / NEXT 29
GET 29
GOSUB / RETURN 30
GOTO 30
IF...THEN 30
IF...THEN...ELSE 31
INPUT 31
INPUT...AT 31
LET 31
LINE INPUT 32
LINE INPUT...AT 32
MOVE 33
NEXT 33
NOTE 33
ON ERROR 33
ON...GOSUB / RETURN 34
ON...GOTO 34
OPEN 35
OPTION BASE 36
OPTION CHR1,OPTION CHR2,OPTION CHR0 36
OPTION PLM1, OPTION PLM2, OPTION PLM0 37
OPTION RESERVE 37
PRINT 37
PRINT...AT 38
PRINT...SPC 38
PRINT...TAB 38
PRINT USING 39
PUT 42
RANDOMIZE 42
READ / DATA 43
REM oder ! oder ' 43
RESTORE 44
RESUME 44
RETURN 45
STACK 45
STOP 45
VARPTR 45
WAIT...AND 46

5.
PROGRAMM FUNKTIONEN 47
NUMERISCHE FUNKTIONEN 47
ATN 47
COS 47
EXP 47
INT 47
LOG 47
RND 47
SGN 48
SIN 48
SQR 48
TAN 48
ALPHANUMERISCHE (STRING-) FUNKTIONEN 48
+ (zieht Strings zusammen) 48
ASC 48
CHR$ 48
INKEY$ 49
INSTR 49
LEFT$ 49
LEN 49
MID$ 49
RIGHT$ 49
SCRN$ 50
STR$ 50
STRING$ 50
TIME$ 51
VAL 51

ATARINSIDE

FUNKTIONEN DIE EINE BESTIMMTE WIRKUNG HABEN 51
EOF 51
ERL 51
ERR 51
FRE 52
PEEK 52
POKE 52
STATUS 52
TIME 53
USR 53

6. BESONDERHEITEN DES ATARI COMPUTER SYSTEMS 55
ÜBERSICHT ÜBER DIE GRAFIKMÖGLICHKEITEN 55
GRAFIKBEFEHLE 56
FARBEN (COLOR) DES ATARI 57
SETCOLORBEFEHL 59
PLOT / PLOT...TO 59
FILL 60
CLS 60
DIE GERÄUSCH- (SOUND-) BEFEHLE 64
STEUERMÖGLICHKEITEN 66
DREHREGLER (PADDLE) 67
STEUERKNÜPPEL (JOYSTICK) 67
SPEZIELLE TASTEN DER COMPUTERTASTATUR 68

7. PLAYER-MISSILE GRAFIK 69
Wie hilft MICROSOFT BASIC II PLAYER-MISSILE Grafiken zu erstellen ? 69
Entwurf eines Spielers (PLAYERS) auf Papier 71
Farbenbestimmung 71
Größenfestlegung 72
Ausgangspositon und Bewegung 72
Vorrang festsetzen 73
Berührung feststellen 73
Demonstrationsprogramm 74

ANHANG 78
A Beispielprogramme 78
B Programme für verschiedene Grafikstufen 85
C Alternative Zeichensätze 87
D Eingangs- / Ausgangsstruktur (INPUT/OUTPUT DEVICE 91
E was liegt wo im Speicherbereich 92
F Konvertierung bestehender Programme 104
G Konvertierung in COMMODORE BASIC 4.0 geschriebenen Programmen 105
H Konvertierung von TRS-80 Radio Shack Programmen in ATARI Microsoft BASIC II 107
I Konvertierung von APPLESOFT - Programmen in ATARI Microsoft BASIC II 109
J Konvertierung ATARI 8k BASIC in Microsoft BASIC II 110
K Der interne Zeichensatz des ATARI (ATASCII) 112
L Verwendung der CIOUSR-Routine 122
M Bei Beendigung eines Programms 126
N Alphabetische Aufstellung der ATARI Microsoft BASIC II Programmierworte 127
O Bedeutung der Fehlermitteilungen 136

INHALTSVERZEICHNIS 139

ATARINSIDE

# VORWORT

In diesem Handbuch finden Sie alle Befehle und Programmierworte, die beim ATARI Microsoft BASIC II benutzt werden können. Auf der ersten Seite des Handbuchs sind diese noch einmal in alphabetischer Reihenfolge mit Angabe der jeweiligen Seitenzahl aufgeführt.

BASIC wurde entwickelt auf dem Dartmouth College von John Kemeny und Thomas Kurtz. Damit wurde eine Computersprache entwickelt, die einfach erlernbar ist und in der Anwendung leicht gehandhabt werden kann. Durch ständige Verbesserungen wurde BASIC zur meistbenutzten höheren Programmiersprache, die auch dem erfahrenen Programmierer kaum Wünsche offen läßt.

Dieses Handbuch ist als Referenzmaterial zur Benutzung mit dem ATARI Heim Computer System beigelegt. Es ist nicht zum Erlernen der Programmiersprache BASIC gedacht, sondern zeigt nur die Besonderheiten des ATARI Microsoft BASIC II auf.

ANMERKUNG: Programme, die mit der Diskettenversion des Microsoft BASIC erstellt wurden, laufen ohne Änderungen auch unter Microsoft BASIC II.

ATARINSIDE

# NEUE MöGLICHKEITEN UND WEGE DER PROGRAMMIERUNG

Willkommen in der Welt des ATARI Microsoft BASIC II. Es ist die am weitesten entwickelte Programmiersprache unter den BASIC-Dialekten, die in einem 16k Einsteckmodul für den ATARI Computer angeboten wird. Sie werden feststellen wie vielseitig und schnell diese Programmiersprache arbeitet.

Einer der Vorteile des Microsoft BASIC II besteht in der leichteren Handhabung von alphanumerischen Konstanten / Variablen (STRINGS). Jetzt können Sie numerische mehrdimensionale Felder und Stringvariablen verarbeiten. Die Zeilennummerierung erfolgt auf Wunsch automatisch (AUTO) und ein Befehl zum Löschen bestehender Zeilen besteht auch (DELETE). Wenn Sie möchten, können Sie die Zeilen eines fertigen Programms neu nummerieren lassen (RENUM). Weiterhin haben Sie nun vom BASIC aus die Möglichkeit DOS anzusprechen, ohne dies aufzurufen (u.a. KILL, LOCK, UNLOCK, NAME ...).

Beim Microsoft BASIC II erfolgt kein Syntax Check, d.h. es wird nicht bei der Eingabe der Zeile bereits überprüft, ob diese den Regeln vom Microsoft BASIC II entspricht. Die Fehlerangabe erfolgt in englischer Sprache nach dem Start des Programms mit RUN. Mit weiteren Befehlen können Sie sich anzeigen lassen, welche Zeile gerade vom Programm abgearbeitet wird (TRON, TROFF). Das ATARI Microsoft BASIC II verfügt über eine sehr genaue Fließkommaarithmetik (16 Stellen) und kann drei Variablentypen ansprechen: ganzzahlig, einfach genau und doppelt genau (DEFINT, DEFSNG, DEFDBL). Normalerweise wird mit einfacher Genauigkeit gerechnet. Durch Einsatz des Zeichens % bei ganzzahligen oder durch das Zeichen # in doppelt genauen Variablen kann die Rechengenauigkeit umgewandelt werden.

Sollen fertig geschriebene und halbfertiggestellte Programme zusammengefügt werden, so kann dies mit dem Befehl MERGE (Mischen) geschehen.

Für den fortgeschrittenen Programmierer bietet das Microsoft BASIC II viele Annehmlichkeiten. So werden beispielsweise ganze Speicherbereiche mit einem Befehl verschoben (MOVE). Weiterhin bestehen kurze Grafik- und Soundbefehle und Sonderbefehle wie der Befehl AFTER (nach), der eine bestimmte Zeit abwartet, bis er den nächsten Befehl ausführt. Dann gibt es noch eine Reihe von OPTION Befehlen. Dort werden Speicherbereiche für die Erstellung von Player-Missile Objekten, von Charaktersätzen oder Maschinenspracheunterroutinen bereitgestellt.

ATARINSIDE

Diese Aufzählung kann beliebig weitergeführt werden. Sie werden sehen, wie man eigene Zusatzfunktionen definieren kann und welche Plotfähigkeiten Ihnen offen stehen.

## WIEVIEL SPEICHERPLATZ WIRD BENöTIGT ?

Um das ATARI Microsoft BASIC II benutzen zu können, benötigen Sie einen ATARI Computer mit mindestens 16k RAM und ein Fernsehgerät bzw. einen Monitor. Wenn Sie Programme von / auf Kassette oder Diskette laden / speichern wollen, benötigen Sie zusätzlich einen ATARI Programmrekorder oder ein ATARI Diskettenlaufwerk. Die BASIC II Erweiterungsdiskette kann nur unter Verwendung des ATARI Disk Operating System (DOS), Version 2.0 benutzt werden.

## LADEN DES MICROSOFT BASIC II

### WENN SIE KEIN DISKETTENLAUFWERK BESITZEN:

1. Vergewissern Sie sich, daß Ihr Computer ausgeschaltet ist.

2. Stecken Sie das Einsteckmodul in den dafür vorgesehenen Steckplatz an Ihrem ATARI Computer.

3. Schalten Sie den Computer ein.

4. Das Microsoft BASIC II wird automatisch in den Speicherbereich des Computers eingeladen.

### WENN SIE EIN DISKETTENLAUFWERK BESITZEN :

1. Vergewissern Sie sich, daß Ihr Computer ausgeschaltet ist.

2. Stecken Sie das Einsteckmodul in den dafür vorgesehenen Steckplatz Ihres Computers.

3. Schalten Sie das Diskettenlaufwerk 1 an und warten Sie bis die mit BUSY bezeichnete Leuchtdiode erlischt.

4. Führen Sie die Diskette so in das Diskettenlaufwerk ein, das der Aufkleber nach oben zeigt.

5. Schalten Sie den Computer ein.

6. Nun wird die auf der Diskette enthaltene Erweiterung zum Microsoft BASIC II in den Speicherbereich Ihres Computers eingeladen.



## KONTROLLANGABEN AUF DEM BILDSCHIRM

Wenn das Microsoft BASIC II eingeladen ist, erscheint auf dem Bildschirm folgende Mitteilung :

Ohne Diskettenlaufwerk Mit Diskettenlaufwerk.

Sollte keine Mitteilung erscheinen, schalten Sie Ihren Computer aus und wiederholen Sie die unter Abschnitt "Einladen des Microsoft BASIC II" vorgeschriebenen Schritte.

## KOPIEREN DER MICROSOFT BASIC II ERWEITERUNGSDISKETTE

Auf der Microsoft BASIC II Diskette befinden sich verschiedene Programme, die die Möglichkeiten des Einsteckmoduls erweitern. Diese Dateien (FILES) haben folgende Bezeichnungen: DOS.SYS, DUP.SYS, AUTORUN.SYS, RS232.SYS, CIOUSR und MEM.SAV. Damit diese Dateien (FILES) nicht versehentlich überschrieben werden können, hat die Diskette eine Schreibschutzhülle; (d.h. die Schutzhülle hat keine Einkerbung. Dadurch ist ein Beschreiben der Diskette nicht möglich). Um nun eigene Programme auf Disketten speichern zu können, müssen Sie erst eine Kopie der Original- (MASTER-) Diskette anfertigen.

Dies wird im Folgenden beschrieben:

1. Wenn die o.a. Mitteilung auf dem Bildschirm erschienen ist, geben Sie den Befehl DOS ein und drücken Sie die RETURN-TASTE. (DOS steht für Disk-Operating-System = Disketten-Betriebs-System). Wenn der nun folgende Ladevorgang beendet ist, erscheint das DOS-Menue auf dem Bildschirm.

2. Nehmen Sie die Erweiterungsdiskette aus dem Diskettenlaufwerk.

3. Legen Sie eine leere unformatierte Diskette in das Laufwerk.

4. Der Computer wartet nun auf Ihre Eingaben:

```
Computer (C)     : SELECT ITEM OR RETURN FOR MENU
Sie geben ein (S): 1   RETURN-Taste druecken

C: WHICH DRIVE TO FORMAT?
S: 1   RETURN-Taste druecken


C: TYPE "Y" TO FORMAT DISK 1
S: Y
```

ATARINSIDE

5. Sie hören nun, wie das Diskettenlaufwerk arbeitet. Wenn die Diskette formatiert / initialisiert ist, erscheint wieder die Mitteilung "SELECT ITEM OR RETURN FOR MENU" auf dem Bildschirm.

Drücken Sie die RETURN TASTE damit Sie zum DOS Menü zurückkehren. Dann verfahren Sie wie folgt:

S: J
C: DUP DISK-SOURCE, DEST DRIVES ?

S: 1,1
C: INSERT SOURCE DISK, TYPE RETURN

S: RETURN
C: TYPE "Y" IF OK TO USE PROGRAMM AREA
CAUTION:A "Y" INVALIDATES MEM.SAV

MEM.SAV ist eine Datenablage in die ein Programm geschrieben wird, wenn Sie von BASIC aus das DOS aufrufen. Nach der Rückkehr ins BASIC wird es automatisch wieder eingeladen.

S: Y

Der erste Teilbereich der Diskette wird in den Speicherbereich des Computer eingelesen.

C: INSERT DESTINATION DISK, TYPE RETURN

Nun legen Sie die vorhin formatierte Diskette ein und drücken die RETURN TASTE. Der im Speicher des Computers vorhandene Teilbereich wird auf die eingelegte Diskette geschrieben.

Dieser Vorgang wiederholt sich noch einige Male. Sie werden jeweils aufgefordert die Quellen- (SOURCE-) Disk oder die Bestimmungs- (DESTINATION-) Disk einzulegen und RETURN zu drücken.

Wenn die Mitteilung "SELECT ITEM OR RETURN FOR MENU" erscheint, ist das Kopieren beendet. Geben Sie nun A ein und betätigen Sie zweimal die RETURN TASTE. Es werden alle auf der Diskette vorhandenen Programmnamen (FILES) angezeigt. Für weitere Informationen lesen Sie bitte das ATARI Disketten Operations System Handbuch.

6.Schalten Sie nun Ihren Computer aus und wieder ein. Dadurch wird die von Ihnen erstellte Kopie eingeladen (geBOOTet). Dies ist erforderlich, weil die in der Datei MEM.SAV enthaltenen Daten zerstört wurden.

7. Sonst können Sie vom DOS durch die Eingabe von B RETURN erreichen, daß das Microsoft BASIC II automatisch wieder in den Speicherbereich Ihres Computers geladen wird.

ANMERKUNG: Bei Punkt 7. ist es wichtig, daß diejenige Diskette sich im Laufwerk befindet, von der das DOS geladen wurde. Es kann sonst passieren, daß sich der Computer "aufhängt"(nicht mehr weiterarbeitet, was sich durch die Ausführung des Punktes 6. wieder beheben läßt).



# BEGINN DER PROGRAMMIERUNG

Dieses Handbuch beschreibt alle Möglichkeiten der Programmierung mit Microsoft BASIC II. Es enthält alle Informationen zum Erstellen einfacher oder komplizierter Programme in Microsoft BASIC II.

## DIREKTE UND INDIREKTE EINGABEN

Bei der Programmiersprache BASIC können Sie die Befehle direkt eingeben. Das geschieht durch die Eingabe des Programmierwortes und dem Drücken der RETURN-Taste. Der gegebene Befehl wird dann sofort ausgeführt. Die andere Eingabeart besteht aus der Eingabe mit einer Zeilennummer. Dieser Befehl wird erst nach dem Start des Programms mit RUN (Lauf) ausgeführt. Beim Microsoft BASIC II können Sie Zeilennummern von 0 bis 63999 verwenden.

Das READY- (Bereit-) Zeichen (>) teilt Ihnen mit, daß der Computer auf Ihre Eingaben wartet. Wir wollen folgendes Beispiel eingeben.

>
S: PRINT"DIES IST EINE DIREKTEINGABE."
C: DIES IST EINE DIREKTEINGABE.

Nun ein Beispiel für die indirekte Eingabe eines Befehls:

>
S: 10 PRINT"DIES IST EINE INDIREKTE EINGABE."

Diese Information wird nun vom Computer in den Speicherbereich geschrieben. Die Ausführung erfolgt erst nach der Eingabe von: RUN . Wenn Sie ein RUN ohne nachfolgende Zeilennummer eingeben, beginnt das Programm den Befehl in der ersten Zeilennummer auszuführen.

## PROGRAMMIERWORTE (KEYWORDS)

Der Computer kann nur Befehle ausführen, die er auch versteht. BASIC ist eine Programmiersprache, die sich an englischen Ausdrücken und Begriffen orientiert. Nehmen wir z.B. das Programmierwort PRINT. Mit ihm wird dem Computer mitgeteilt, daß er etwas auf den Bildschirm schreiben soll;(englisch PRINT bedeutet SCHREIBE). Im Microsoft BASIC II werden über 100 dieser Worte verwendet. Sollten Sie ein Wort benutzen, das der Computer nicht kennt, erhalten Sie eine Fehlermitteilung. Natürlich dürfen reservierte Worte nicht zur Benennung von Variablen verwendet werden.

## DIE PROGRAMMZEILE

Wenn Sie ein Programm schreiben möchten, geschieht dies unter Verwendung von Zeilennummern, d.h. Sie müssen vor die Programmierworte eine Ziffer setzen. Zweckmäßigerweise beginnen Sie mit 10 und lassen für spätere Einfügungen einen Abstand von 10 zwischen den Zeilen. Ein Beispiel:

10 LET A=10
20 PRINT A

Die Eingabe der Befehle muß in Großbuchstaben erfolgen.

## DIE BEDEUTUNG DER INTERPUNKTION

Anders wie bei der Rechtschreibung haben Punkte, Komma, Semikolon und ähnliche Interpunktionen im Microsoft BASIC II eine Befehlsfunktion. Nachfolgend sind die Art und Bedeutung dieser Befehle aufgeführt.

ATARI INSIDE

Zwischenraum

Beim Microsoft BASIC II ist es erforderlich vor und nach jedem Programmierwort eine Leerstelle zu setzten. Dies geschieht durch Betätigen der Leertaste. Sollten Sie diese Zwischenräume vergessen, erhalten Sie beim Programmstart die Fehlermitteilung SYNTAX ERROR.

Anführungszeichen

Bei der Rechtschreibung wird die wörtliche Rede in Anführungszeichen gesetzt. Ähnlich ist es beim Microsoft BASIC II. Sie bestimmen den Anfang und das Ende des zu schreibenden Textes. Ein Beispiel:

```
S: 10 PRINT "DIES IST EIN BEISPIEL FUER DIE"
S: 20 PRINT ""VERWENDUNG VON ANFUEHRUNGSZEICHEN""
S: RUN
```

Durch die Eingabe doppelter Anführungszeichen wird der Text in Zeile 20 bei der Ausgabe mit Anführungszeichen ausgegeben.

Kommas

Das Komma hat drei Funktionen:

1. Es dient der Unterscheidung verschiedener Parameter, die bei bestimmten Befehlen benötigt werden. Zum Beispiel beim SOUND-Befehl. Dieser Befehl wird eingegeben als SOUND 2,&79,10,8,60. Dabei wird festgelegt, daß der zweite Soundgenerator einen Ton mit der Höhe von hexadezimal 79, einer Verzerrung von 10 und einer Lautstärke von 8 ausgeben soll. Der Ton soll 60 Fünfzigstelsekunden dauern.

2. Weiterhin werden Kommas benutzt um mehrere Variable mit einem INPUT Befehl eingeben zu können. Zum Beispiel INPUT A,B,C,D,E. Dies erspart Platz beim Programmieren und schafft Übersicht bei der Auflistung. Der Computer erwartet bei dem angeführten Beispiel die Eingabe von fünf Variablen.

3.Zuletzt ist das Komma beim PRINT-Befehl zum Absetzen bestimmter Textteile erforderlich. Nehmen wir an, Sie möchten das Ergebnis einer Berechnung ausgeben. X hätte in Ihrem Programm den Wert 25. Sie geben ein :20 PRINT "SIE SIND",X,"JAHRE ALT" . Nach der Eingabe von RUN, RETURN wird der Textteil SIE SIND geschrieben. Die Ziffer von X (25) wird auf die 14. Stelle geschrieben. Weitere 14 Stellen weiter erscheint die Mitteilung JAHRE ALT.



Semikolons

Das Semikolon wird ausschließlich in Verbindung mit dem PRINT-Befehl benutzt. Mit ihm ist eine Verkettung von verschiedenen Textpassagen oder Variablen möglich. Wenn Sie das Beispiel in der Beschreibung der Kommafunktion wie folgt umändern, sehen Sie es am praktischen Beispiel:

```
S: 10 PRINT "SIE SIND ";X;" JAHRE ALT"
```

Nach dem Start des Programms mit RUN,RETURN erscheint dann folgender Text: SIE SIND 25 JAHRE ALT. Diese Funktion ist besonders nützlich wenn die Variable 2-,3- oder noch mehrstellig ist.

Eine weitere Möglichkeit ergibt sich, wenn Sie ein Semikolon hinter einen zu schreibenden Text setzen. Beim nächsten PRINT-Befehl, wird dieser an der Stelle ausgeführt. Aber auch hierzu ein Beispiel:

```
10 PRINT "DER WERT IST ";
20 WERT=123.45
30 PRINT WERT
S: RUN
C: DER WERT IST 123.45
```

Doppelpunkt

Mit dem Doppelpunkt werden Befehle voneinander getrennt, die unter einer Zeilennummer zusammengefaßt werden. Es ist nämlich nicht erforderlich, jedem Befehl eine eigene Zeilennummer zu geben. Sie können bis zu drei Bildschirmzeilen unter einer Zeilennummer eingeben.

```
10 X=5:Y=3:Z=X+Y:PRINT Z:END
```

Für diese eine Zeile hätten Sie sonst 5 Programmzeilen benötigt.

Verbessern (Editieren) bestehender Programme oder Programmteile.

Die ATARI Computer bieten einige komfortable Möglichkeiten zur Verbesserung / Bearbeitung von Texten, Programmen usw. Nach dem Einschalten des Systems können Sie nur große Buchstaben schreiben. Durch Betätigen der mit CAPS LOWR bezeichneten Taste schalten Sie auf Kleinschreibung um. Nun ist die Bedienung des ATARI wie die einer Schreibmaschine. Mit den SHIFT-Tasten wird umgeschaltet zwischen Groß- und Kleinschreibung. Durch nochmaliges Betätigen der CAPS LOWR Taste wird sinnbildlich die SHIFT-Taste wieder verriegelt.(Bei ATARI 400 und 800 müssen zum Verriegeln die SHIFT- und die CAPS LOWR-Taste gleichzeitig gedrückt werden.)

ATARINSIDE

## Die Kontroll- und SHIFT-Tasten

Die Cursor Kontroll-Tasten erlauben ein schnelles Editieren des auf dem Bildschirm ausgegebenen Textes. Diese Sondertasten arbeiten entweder in Verbindung mit der CTRL (Control) -Taste oder zusammen mit der SHIFT-Taste. Das bedeutet, daß beide Tasten gleichzeitig gedrückt werden müssen.Nachfolgend sind die Tastenkombinationen und ihre Funktionen aufgeführt:

CTRL und ↑ bewegt den Cursor nach oben
CTRL und → bewegt den Cursor nach rechts
CTRL und ↓ bewegt den Cursor nach unten
CTRL und ← bewegt den Cursor nach links

Diese Funktionen werden ausgeführt, ohne das etwas im angezeigten Text bzw. Programm verändert wird. Wenn der Cursor aus dem Bildschirm geführt wird, erscheint er auf der gegenüberliegenden Seite wieder.

CTRL und INSERT fuegt vor dem unter dem Cursor stehendem einen Zwischenraum ein .

CTRL und DELETE bewegt die rechts vom Cursor stehende Zeichen eine Stelle nach links. Dabei wird das unter dem Cursor stehende Zeichen gelöscht.

CTRL und 1 läßt das Bild auf dem Bildschirm "einfrieren", d.h. das ein Listing oder ein Programm durch diese Tastenkombination angehalten werden kann. Werden die Tasten erneut betätigt, wird der Ablauf fortgesetzt.

CTRL und 2 läßt einen Kontrollton ertönen.

Wenn Sie die SHIFT-Taste gedrückt lassen, werden von den doppelt belegten Tasten die oberen Symbole eingegeben.

SHIFT und INSERT verschiebt die Zeile, in der sich der Cursor befindet um eine Zeile nach unten. Somit ist Platz zum Schreiben einer neuen Zeile.

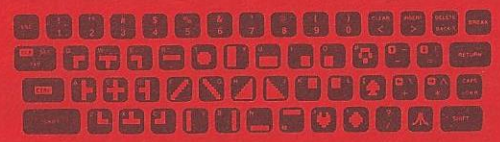
SHIFT und DELETE löscht die Zeile, in der sich der Cursor befindet.

Der Begriff der Zeile bezieht sich hierbei auf eine tatsächliche Bildschirmzeile, nicht etwa auf eine gesamte Programmzeile.

## GRAFISCHE SYMBOLE

Die CTRL-Taste funktioniert wie eine zusätzliche SHIFT Taste. Wenn die CTRL-Taste zusammen mit einer Buchstabentaste gedrückt wird, erschließen sich dem Anwender die Grafikzeichen des ATARI Computers. Sie können zum Erstellen einfacher Zeichnungen, Diagrammen o.ä. verwendet werden. Auf der nächsten Seite sehen Sie, welche Grafikzeichen welchem Buchstaben zugeordnet sind.





## TASTEN MIT SPEZIELLEN FUNKTIONEN

### INVERS VIDEO TASTE

Inverse Video (umgekehrte Darstellung).
Durch Betätigung dieser Taste werden die danach eingegebenen Zeichen invers dargestellt. Sie sind dann nicht weiß auf blauem Untergrund sondern blau auf weißem Untergrund. Nochmaliges Drücken dieser Taste schaltet die Funktion wieder aus.

### CAPS LOWR-Taste

Durch Druck auf diese Taste wird die Kleinschreibung eingeschaltet. Nochmaliges Drücken schaltet wieder auf Großschreibung um. (Wichtig zur Programmierung. Die Programmierworte müssen groß geschrieben werden, damit sie der Computer erkennen kann). Bei den Modellen ATARI 400 / 800 erfolgt das Zurückschalten durch gleichzeitiges Drücken der SHIFT- und CAPS LOWR-Tasten.

### ESC-Taste

Diese Taste bedeutet Escape, was in mit Flucht übersetzt werden kann. Mit ihr werden spezielle Funktionen erfüllt. Dies läßt sich am besten mit einem Beispiel erklären. Im Direkteingabemodus drücken Sie zum Löschen des Bildschirms gleichzeitig die Tasten CTRL und CLEAR. Um diesen Befehl im Programm ausführen zu lassen müssen Sie eingeben:

```
10 PRINT "ESC CTRL CLEAR" RETURN
```

Dabei werden die CTRL- und CLEAR-Tasten wieder gleichzeitig gedrückt. Nun wird der Bildschirm immer gelöscht, wenn die Zeile 10 abgearbeitet wird.

### BREAK-Taste

Diese Taste unterbricht (Break) die Programm- bzw Befehlsausführung z.B. den Befehl LIST. Es erscheint das READY-Symbol ( > ). Durch die Eingabe von CONT und RETURN wird in den meisten Fällen die Programmausführung wieder aufgenommen .

### SYSTEM RESET-Taste

Mit dieser Taste wird das Computersystem wieder in seine Ausgangsposition zurückgesetzt.(System Reset = System rücksetzen). Der Bildschirm wird in Grafikmodus 0 geschaltet, und gelöscht. Im Prinzip entspricht das zurücksetzen einem aus- und wieder einschalten des Computers. Nur das Programm im Speicher bleibt erhalten und kann mit RUN RETURN erneut gestartet werden kann.

ATARI INSIDE

CLR SET TAB-Taste

Mit dieser Taste können Sie Tabulatormarken setzen. Dazu drücken Sie die SHIFT- und die CLR SET TAB-Taste gleichzeitig. An der Stelle, an der sich der Cursor befindet wird eine Tabulatormarke gesetzt. Zum Löschen der Tabulatormarken drücken Sie die CTRL- und die CLR SET TAB-Taste gleichzeitig.

Bei jedem Druck auf die Taste springt nun der Cursor an die nächste festgelegte Stelle des Bildschirms. Dies ist bei der Erstellung von Tabellen o.ä. sehr nützlich. Der Tabulator kann auch vom Programm aus gesetzt / gelöscht werden. Dies geschieht unter Verwendung der ESC-Taste:

```
10 PRINT "ESC   SHIFT   CLR SET TAB"
20 PRINT "ESC   CTRL    CLR SET TAB"
```

Beim Einschalten des Computers stehen die Tabulatormarken auf den Werten 7, 15, 23, 31 und 39.

RETURN-Taste

Wie Sie schon an den Programmbeispielen gesehen haben, wird diese Taste benötigt um Eingaben abzuschließen. Dabei ist es gleichgültig, ob es sich um eine Eingabe im Direktmodus handelt oder um den Abschluß einer Programmzeile.



# BESTANDTEILE EINES PROGRAMMS

KONSTANTEN UND VARIABLEN

Die Konstanten sind Zahlen oder Buchstaben die in einem Programm benutzt werden.
Sie behalten während des gesamten Programmablaufs ihren Namen oder Wert. Beispiele: 5 oder "HANS".

Variablen sind Zahlen oder Buchstaben, die sich während des Programmablaufes ständig ändern bzw. geändert werden. Beispiele: A oder J$ .

Microsoft BASIC II benutzt fünf Arten von Konstanten und Variablen: ganzzahlige, solche mit einfacher / doppelter Genauigkeit, hexadezimale und Strings.

BILDEN EINES VARIABLENNAMENS

Zur Bildung einer Variablen dürfen alle Buchstaben von A-Z und alle Zahlen von 0-9 verwendet werden. Auch die Unterstreichung (_) ist erlaubt. Bei der Festlegung des Namens muß zunächst ein Buchstabe eingegeben werden. So wäre beispielsweise A9 ein zulässiger Name, bei 9A dagegen würde eine Fehlermitteilung erscheinen.

SPEZIFIZIEREN DER GENAUIGKEIT NUMERISCHER VARIABLEN

Sie können den Typ und damit die Genauigkeit der Variablen auf zwei Wegen bestimmen.

1. Sie geben einen der Befehle DEFINT, DEFSNG, DEFDBL oder DEFSTR ein.

2. Sie geben hinter der Variablen eines der Symbole zur Identifizierung ein (%,#,$).

Die unter Punkt 1 aufgeführten Befehle können Sie überall im Programm benutzen. Es ist jedoch sinnvoll, sie an den Anfang zu stellen.

GANZZAHLIGE KONSTANTEN
Beispiele: 23, -999,709,32000

Alle ganzen Zahlen zwischen -32768 und 32767 werden in zwei Bytes abgegelegt. Dabei wird bei einer Multiplikation eine Zahl einfacher Genauigkeit errechnet. Bei anderen Rechenoperationen wird das Ergebnis mit doppelter Genauigkeit errechnet.

GANZZAHLIGE VARIABLEN
Beispiele: DMARK%, X%, SUMME%

Durch das Anfügen des Prozentzeichens ( % ) wird die Variable als ganzzahlig deklariert.

ATARINSIDE

DEFINT (bestimmen der Variablen als ganzzahlig)
Format : DEFINT (Buchstabe),(von Buchstabe...-bis Buchstabe...)
Beispiele: 10 DEFINT N,J,K-M
20 DEFINT I

Durch diesen Befehl werden alle Variablen deren Namen mit einem der aufgeführten Buchstaben beginnen, als ganzzahlige Variablen festgelegt. Die Zeilen sollten am Anfang eines Programms stehen. Trotzdem koennen Sie eine oder mehrere bestimmte Variable mit einem Wert doppelter Genauigkeit bestimmen, wenn Sie hinter den Namen ein Nummerzeichen (#) setzen.

Der Wert der ganzen Zahl muß zwischen -32768 und 32767 liegen. Der Vorteil in der Verwendung einer ganzen Zahl liegt in der großen Verarbeitungsgeschwindigkeit.

KONSTANTEN MIT EINFACHER GENAUIGKEIT
Beispiele: 65E12,333335,,45E8,,33E-6

Alle Konstanten, die nicht ganze Zahlen zwischen -32768 und 32767 sind, werden als Konstanten mit einfacher Genauigkeit festgelegt.

VARIABLEN MIT EINFACHER GENAUIGKEIT
Beispiele: AMT, LAENGE, SEITE

Wenn die Variablen nicht anders bestimmt werden, werden sie mit einfacher Genauigkeit festgelegt, d.h. die Zahl wird auf 6 Stellen errechnet. Der Exponent (Hochzahl) kann sich im Bereich von -38 bis +38 bewegen.

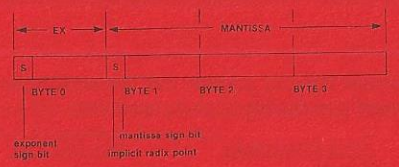
DEFSNG (bestimmen der Variablen mit einfacher Genauigkeit)
Format :DEFSNG (Buchstabe),(von Buchstabe...-bis Buchstabe...)
Beispiele: 10 DEFSNG K,S,A-F
20 DEFSNG Y

Alle Variablen, die mit den angegebenen Buchstaben beginnen, werden als Werte mit einfacher Genauigkeit erkannt und verarbeitet. Dazu zählen auch die Variablen von A bis einschließlich F. Die Namen der Variablen müssen durch ein Komma voneinander getrennt werden.
Beispiel Programm:

```
10 DEFSNG A-F
20 DM=DM+1
30 PRINT DM
40 GOTO 20
```

In diesem Beispielprogramm werden alle Variablen von A bis F als Zahlen mit einfacher Genauigkeit definiert (bestimmt). Da die Variable DM mit einem D beginnt, gilt das auch für diese. Wird irgendwo im Programm die Variable als DM # angegeben, erfolgt an dieser Stelle die Berechnung mit doppelter Genauigkeit. Im folgenden Bild wird die Darstellung dieses Variablentypes im Speicher verdeutlicht.






KONSTANTEN MIT DOPPELTER GENAUIGKEIT
Beispiele: 45D5, 23D-6, 8888888D-12

Sie können Konstanten als doppelt genaue Konstanten bestimmen, indem Sie vor den Exponenten den Buchstaben D statt wie üblich E setzen. Eine doppelt genaue Ziffer wird in 8 Bytes abgelegt. Die Verarbeitung erfolgt auf 16 Stellen genau.

VARIABLEN MIT DOPPELTER GENAUIGKEIT
Beispiele: DBL#,X#,DM#

Durch den Zusatz des Nummernzeichens ( # ) wird die Variable als doppelt genaue Variable bestimmt. Sie wird in 8 Bytes abgelegt. Dabei werden der Exponent und sein Vorzeichen im ersten Byte festgelegt. Der Exponent kann von -38 bis +38 reichen. Die anderen 7 Bytes enthalten die Mantisse und ihr Vorzeichen.

DEFDBL (bestimmen der Varaiblen mit doppelter Genauigkeit)
Format : DEFDBL (Buchstabe),(von Buchstabe...-bis Buchstabe...)
Beispiele: 10 DEFDBL Z,C-E
20 DEFDBL R

Wie bei der Bestimmung der einfachen Genauigkeit, werden alle Variablennamen, die mit einem der im Beispiel angeführten Buchstaben beginnen, als solche mit doppelter Genauigkeit erkannt und verarbeitet. In Bild 2-3 wird die Darstellung dieses Variablentypes im Speicher dargestellt.

HEXADEZIMALE KONSTANTE
Beispiele: &76, &F3, &7B, &F3EB

Oft ist es erforderlich hexadezimale Zahlen in einem Programm zu verwenden. Sei es als Adresse im Speicherbereich oder zum Erstellen kurzer Maschinenspracheunterroutinen mittels DATA-Anweisungen. Dazu mußte man bisher die Hexzahl erst in eine Dezimalzahl umrechnen. Dies entfällt beim Microsoft BASIC II. Hier können nach Voranstellung des &-Zeichens direkt Hexzahlen eingegeben werden.

Startet eine Maschinenspracheroutine an der Speicheradresse hexadezimal C305, so können Sie A=USR(&C305,0)eingeben. Bei dem Befehl A=PEEK(&5A61) wird der Variablen A der Inhalt der Speicherzelle hexadezimal 5A61 zugewiesen. Sollten Sie eine Umrechnung wünschen, geben Sie PRINT & ein (und die Hexzahl). Nach Drücken der RETURN-TASTE erhalten Sie das dezimale Ergebnis. Besonders für graphische Anwendungen (z.B. Player-Missile-Grafik) erweist sich die hexadezimale Darstellung als vorteilhaft.

In der folgenden Tabelle werden die Umrechnungen zwischen Dezimalzahlen, Hexadezimalzahlen und Binärzahlen dargestellt:

| DEZIMAL | HEXADEZIMAL | BINARZAHL |
|---|---|---|
| 01 | 01 | 0001 |
| 02 | 02 | 0010 |
| 03 | 03 | 0011 |
| 04 | 04 | 0100 |
| 05 | 05 | 0101 |
| 06 | 06 | 0110 |
| 07 | 07 | 0111 |
| 08 | 08 | 1000 |
| 09 | 09 | 1001 |
| 10 | A | 1010 |
| 11 | B | 1011 |
| 12 | C | 1100 |
| 13 | D | 1101 |
| 14 | E | 1110 |
| 15 | F | 1111 |

ALPHANUMERISCHE-VARIABLEN (STRINGS) UND VARIABLENFELDER (MATRIZEN)
ALPHANUMERISCHE (STRING-) KONSTANTEN
Beispiele: "BLUME", "Geben Sie den Namen ein:_____"

Alphanumerische Konstanten müssen immer in Anführungsstrichen stehen. Sie können eine Maximallänge von 120 Zeichen haben. Sollten Sie mehr Zeichen benötigen ist es möglich zwei Strings zusammenzufügen. Der Begriff alphanumerisch bedeutet, daß Sie alle Zeichen der Tastatur verwenden können, also auch Ziffern, Buchstaben und Sonderzeichen (!"$%&/§(). Auch das Setzen doppelter Anführungszeichen ist erlaubt:""HAUS"". Bei der Ausgabe wird nun der Begriff HAUS in Anführungszeichen gesetzt:"HAUS". Die Zeichen vertikaler Strich (SHIFT =) und das Herz (CTRL ,) bedeuten das Ende einer alphanumerischen Konstanten. Werden diese Zeichen also verwendet, so wird der String an dieser Stelle abgeschnitten.
Geben Sie folgendes Beispiel ein:

```
10 PRINT "Alle Buchstaben und ""Sonderzeichen""";
20 A$="sind beim String erlaubt"
30 PRINT A$
RUN RETURN
```

ALPHANUMERISCHE (STRING-) VARIABLEN
Beispiele: A$, NAME$, ADRESSEN$

Alphanumerische Variable werden durch das Dollarzeichen ( $ ) bestimmt. Sie können eine Länge bis zu 120 Zeichen haben und auch hier ist die Verwendung der doppelten Anführungszeichen erlaubt. Ein Beispiel:

```
10 A$="einige Zeichen"
20 A$="einige andere ""Zeichen"""
```

Sie sehen also, warum dies Variablen sind. Man kann ihnen während des Programmablaufs ständig andere Zeichenfolgen zuordnen.

DEFSTR (bestimmen der Stringvariablen)
Format : DEFSTR (Buchstabe), (von Buchstabe...-bis Buchstabe...)
Beispiele: 10 DEFSTR A,K-M,Z
20 DEFSTR F,J,I,O



Ein String kann als Variablenname angelegt werden. Dazu wird dieser Befehl benutzt. Probieren Sie folgendes Beispiel:

```
10 DEFSTR A
20 A="Ihr Name ist"
30 PRINT A
```

Natürlich ist auch diese Form möglich:

```
10 DEFSTR A
20 A$="Ihr Name ist"
30 A=A$
40 PRINT A
```

VARIABLENFELDER (MATRIZENBILDUNG)(ARRAYS)

Ein Variablenfeld ist eine Liste indizierter Variablen mit dem gleichen Variablennamen und einer in Klammer gesetzten Nummer. Die Indizes dürfen die Werte zwischen 0 und dem in einer DIM-Anweisung festgelegten Wert annehmen. Ohne Dimensionierung sind Indizes bis 10 erlaubt. Dies ist ein eindimensionales Variablenfeld mit 7 Elementen.

A(0)
A(1)
A(2)
A(3)
A(4)
A(5)
A(6)

Beachten Sie, daß hier die Variable A(0) als das erste Element zählt. Geben Sie folgendes Beispiel ein:

```
100 A(1)=55
110 A(2)=77
120 A(3)=93
130 A(4)=61
140 FOR I = 1 TO 4
150 PRINT A(I)
160 NEXT
180 END
```

ATARINSIDE

Ein mehrdimensionales Variablenfeld ist eigentlich nur eine Reihe eindimensionaler Felder. Es kann auch mit einer Matrix verglichen werden. Dabei wird im BASIC immer zuerst die Zeile angegeben. Nachfolgend ist eine 7x4-Matrix zum besseren Verständnis aufgezeigt:

```
                    SPALTEN
Z   A(0,0)    A(0,1)    A(0,2)    A(0,3)
E   A(1,0)    A(1,1)    A(1,2)    A(1,3)
I   A(2,0)    A(2,1)    A(2,2)    A(2,3)
L   A(3,0)    A(3,1)    A(3,2)    A(3,3)
E   A(4,0)    A(4,1)    A(4,2)    A(4,3)
N   A(5,0)    A(5,1)    A(5,2)    A(5,3)
    A(6,0)    A(6,1)    A(6,2)    A(6,3)
```

## ARITHMETISCHE, VERGLEICHS- UND LOGISCHE OPERATIONEN

### ARITHMETISCHE OPERATIONEN

Für Berechnungen stehen folgende Zeichen zur Verfügung:
(), =, -, ^, *, /, +

Diese Symbole können auch mit den anderen Operatoren gemischt eingegeben werden. Zum Beispiel ist der Vergleich A/C > D*A erlaubt. Im Einzelnen bedeuten die Symbole:

() Klammerfunktion (um die Berechnungsfolge zu beeinflussen)
= Gleichheitszeichen
- die diesem Symbol folgende Zahl ist negativ (z.B. -3)
^ Exponentenfunktion (Z^3 bedeutet zwei hoch drei)
* Multiplikation
/ Division
+ Addition
- Subtraktion

### VERGLEICHSOPERATIONEN

In der folgenden Tabelle sind die Symbole und deren Bedeutung aufgeführt:

= gleich. Vergleicht ob zwei Variable gleich sind.
<> ungleich. Das Gegenteil von =
< kleiner als.
> größer als.
<= kleiner als oder gleich.
>= größer als oder gleich.

Diese Vergleiche werden oft im Zusammenhang mit einem IF...THEN benutzt. Interessant ist hier die Verwendung der Boolschen Formel. Dazu Beispiele:

```
10 A=10:B=20
20 IF A=B THEN GOTO 40
30 IF A<>B THEN END
40 PRINT "A und B gleich. Vergleich trifft zu."
```

Boolsche Formel:

```
10 A=33
20 B=(A=29)+(A=30)+(A=31)+(A=32)+(A=33)
30 IF B=1 THEN GOTO 50
40 END
50 PRINT "VERGLEICH TRIFFT ZU."
```

Durch Verändern der Werte in Zeile 10 können Sie die Vergleichsoperationen kennenlernen.



Eine Besonderheit ist der Vergleich von alphanumerischen Variablen. So ist folgende Programmzeile durchaus erlaubt:

```
1020 IF A$ < B$ THEN GOTO 2000
```

Dabei wird zunächst das erste Zeichen in den ATASCII-Code umgewandelt. Dort ist jedem Zeichen eine Ziffer zugeordnet. (z.B. ist A als 65, B als 66 usw. definiert). Dadurch ist ein Vergleich möglich, weil die Ziffern verglichen werden können.

Beispielprogramm:

```
10 INPUT A$,B$
20 IF A$ < B$ THEN 60
30 C$=A$
40 A$=B$
50 B$=C$
60 PRINT A$,B$
70 END
```

Geben Sie zwei Worte ein, die durch Komma getrennt werden. Sie können zwei Worte eingeben, die sich unwesentlich unterscheiden, wie z.B. KLAUSE und KLAUS. Sie werden sehen, daß diese Worte alphabetisch sortiert ausgegeben werden.

### LOGISCHE OPERATIONEN

Die logischen Operationen werden in der Reihenfolge ihrer Prioritäten aufgeführt:

NOT (NICHT-Verknüpfung). Bitweises komplementieren eines Bytes. Ist ein Bit 1, ist es nach der Operation 0.

AND (UND-Verknüpfung). Ein Byte wird bitweise undiert.

OR (ODER-Verknüpfung). Ein Byte wird bitweise oderiert.

XOR (EXKLUSIV ODER-Verknüpfung). Ein Byte wird bitweise exclusiv oderiert.

Die logischen Vergleiche können auch in Verbindung mit alphanumerischen Variablen verwendet werden.(Siehe Seite 48).

Anmerkung: die vergleichenden und logischen Operationen können miteinander kombiniert werden. So ist es z.B. möglich, den folgenden Vergleich zu machen: A > B AND C < D. Zum besseren Verständnis, sollten Sie diese Operationen in einem kleinen Programm ausprobieren.

ATARINSIDE

# 3 PROGRAMM BEFEHLE

AUTO
(Nur mit Erweiterungsdiskette).

Die nachfolgend aufgeführten Befehle werden meist im Direktmodus eingegeben, also nicht mit Zeilennummer innerhalb eines Programms.

Format : (erste Zeilennummer, Schrittweite)
Beispiele: AUTO 200,20
AUTO

Mit diesem Befehl werden die Zeilen automatisch nummeriert. Wenn keine Startzeilennummer angegeben wird, beginnt die Nummerierung mit 100 und erhöht sich nach jedem RETURN um 10.

Sonst beginnt die automatische Nummerierung mit der Startzeilennummer und erhöht sich um die hinter dem Komma angegebene Ziffer. Zum Verlassen des AUTO Modus drücken Sie die RETURN Taste nach Erscheinen der nächsten Zeilennummer oder die BREAK Taste.
Beispielprogramm:

```
AUTO 300,20
300 PRINT "Dies ist ein ";
320 PRINT "Beispiel fuer ";
340 PRINT "AUTO."
360 RETURN
```

ANMERKUNG: Sollte bei der automatischen Nummerierung eine Zeilennummer auftreten, die bereits im Programm existiert, so wird diese automatisch gelistet.

CLOAD (Kassette laden)

Format : CLOAD
Beispiele: CLOAD
100 CLOAD

Hiermit wird das Programm das sich auf der Kassette befindet, in den Speicher des Computers geladen. Geben Sie CLOAD ein und drücken Sie RETURN. Ein Summton ertönt einmal. Positionieren Sie das Tonband in der Kassette so, daß der Anfang des zu ladenden Programms vor dem Tonkopf steht. Benutzen Sie hierzu das Bandzählwerk als Hilfe. Dann drücken Sie die PLAY- (Start-) Taste des Rekorders. Anschließend betätigen Sie die RETURN-Taste. Das Programm wird geladen. Weitere Hinweise entnehmen Sie bitte dem Bedienungshandbuch Ihres Programmrekorders.



CONT (weiter)

Format :CONT

Mit CONT wird das Programm fortgesetzt, wenn es durch den Befehl STOP, durch Drücken der BREAK Taste oder durch einen auftretenden Fehler unterbrochen wurde. Dieser Befehl ist sehr nützlich beim Testen von Programmen. Z.B. können Sie durch den Befehl STOP das Programm unterbrechen. Da die Werte aller Variablen und Konstanten erhalten bleiben, können Sie sich diese auslisten lassen: (PRINT A$ usw.). Danach geben Sie CONT ein und drücken die RETURN-Taste.

CSAVE (auf Kassette speichern)

Format :CSAVE
Beispiele:CSAVE
100 CSAVE

Mit CSAVE wird das Programm, daß sich gerade im Speicher des Computers befindet, auf Kassette abgespeichert. Dadurch steht es auch nach dem Ausschalten des Computers noch zur Verfügung.

Nachdem Sie CSAVE eingegeben haben, drücken Sie die RETURN Taste. Der Summton ertönt zweimal. Dadurch sollen Sie daran erinnert werden, daß Sie nun zwei Tasten am Rekorder drücken müssen, nämlich die RECORD- (Aufnahme-) Taste und gleichzeitig die PLAY- (Start-) Taste. Dann betätigen Sie erneut die RETURN-Taste. Natürlich muß das Tonband vorher an eine unbespielte Stelle gespult werden.
Programme, die mit CSAVE abgespeichert wurden, lassen sich schneller laden als die mit SAVE"C:" gespeicherten. Bedenken Sie jedoch, daß CSAVE und CLOAD zusammen gehören, ebenso SAVE"C:" und LOAD"C:". Ein Vermischen führt zu einer Fehlermeldung.

DEL
(Nur in Verbindung mit Erweiterungsdiskette)

Format : DEL (Startnummer, Endnummer)
Beispiele: DEL 450-
DEL 250-350
DEL -250

Mit DEL (delete / löschen) werden bestimmte Zeilennummern aus dem Speicherbereich des Computers gelöscht. Dabei haben Sie folgende Möglichkeiten:

DEL n löscht die Zeile mit der angegebenen Nummer n.
DEL - n löscht alle Zeilen bis einschließlich Zeilennr. n.
DEL n - löscht Zeilennr. bis einschließlich letzte Zeilennr. n.
DEL n - m löscht Zeilen ab n bis einschließlich m.

Beispielprogramm

```
100 PRINT "Ein Beispiel ";
110 PRINT "fuer den DEL ";
120 PRINT "Befehl."
```

ATARINSIDE

DEL 110- RETURN

Nach dieser Eingabe ist nur noch die Zeile 100 vorhanden. Geben Sie ein:
LIST (RETURN-Taste)
Zum löschen einer einzelnen Zeile können Sie auch eingeben:
100 (RETURN-Taste)
Dadurch wird in diesem Beispiel die Zeile 100 auch gelöscht.
ANMERKUNG: Wenn Sie den DEL-Befehl innerhalb eines Programms verwenden, also nicht im Direktmodus, wird der Programmablauf nach Ausführung des DEL-Befehls gestoppt.

DOS

Format :DOS
Beispiel:DOS

Wenn Sie den DOS-Befehl eingeben wird das Microsoft BASIC II verlassen und das Disketten Operations System geladen. Dadurch wird das im Speicher stehende Programm gelöscht, wenn Sie keine MEM.SAV-Datei (File) auf der Diskette haben. Lesen Sie hierzu auch das Handbuch Ihres Diskettenoperationssystems.

ANMERKUNG: Wenn Sie kein Diskettenlaufwerk besitzen, kommen Sie durch Eingabe des Befehls in den MEMO PAD-Mode (bei den neuen ATARI Modellen kommen Sie zum Selbsttestprogramm).

KILL

Format :KILL "Gerätespezifikation:Programmname"
Beispiel:KILL "D:PROG1.BAS"

Im Beispiel würde das Programm mit dem Namen PROG1.BAS von der Diskette gelöscht werden. Seien Sie vorsichtig mit der Eingabe dieses Befehls, damit Sie nicht versehentlich Programme löschen.

LIST

Format :LIST ("Gerätespezifikation:Programmname)", Nr, No
Beispiele:
```
100 LIST
150 LIST "C:"
200 LIST "P:"
100 LIST "D:TEST.BAS"
150 LIST 100-200
200 LIST -300
```

Mit diesem Befehl werden die Programmzeilen auf dem Bildschirm gelistet. Ohne weitere Zusatzangaben erfolgt die Ausgabe auf dem Bildschirm in steigender Reihenfolge beginnend mit der ersten Programmzeilennummer.
Sind Datenkanäle angegeben, erfolgt die Ausgabe an das jeweilige Gerät:
D: ist Diskette. Hier ist zusätzlich ein Programmname erforderlich.
P: ist Printer, also Drucker.
C: ist Kassette.
E: steht für Editor. Die Zeilen werden auf den Bildschirm gelistet.



Wenn Sie ein Programm auf dem Bildschirm listen, werden Sie feststellen, daß der Bildschirminhalt gescrollt wird. D.h. am unteren Rand des Bildschirm tauchen neue Zeilen auf und die oberen werden aus dem Bild geschoben. Diesen Vorgang können Sie aufhalten, indem Sie die CTRL- und gleichzeitig die mit 1 bezeichnete Taste drücken. Durch erneutes Drücken wird das Bild wieder weitergeschoben.

Nachfolgend sind noch einmal die Möglichkeiten des LIST-Befehls aufgeführt:
LIST Es werden alle Zeilen aufsteigend sortiert gelistet.
LIST n Es wird die Zeile n gelistet.
LIST - n Es werden alle Zeilen einschließlich n gelistet.
LIST n - Es werden alle Zeilen ab n gelistet.
LIST n - m Es werden alle Zeilen von n bis m gelistet.
Beispielprogramm:
```
100 REM Beispiel fuer
110 REM den LIST Befehl
120 PRINT "Es können nun ";
130 PRINT "alle Funktionen ";
140 PRINT "des LIST - Befehls ";
150 PRINT "ausprobiert werden."
```

Eine wichtige Bedeutung kommt dem LIST Befehl in Verbindung mit dem Diskettenlaufwerk zu. Wenn Sie eingeben: LIST"D:(Programmname)", wird das Programm auf Diskette gelistet. Dadurch kann es beim Einladen in ein bestehendes Programm eingefügt werden. (Nützlich für ständig verwendete Unterroutinen u.ä.). Sie können auch bestimmte Programmteile herauslisten, wenn Sie z.B. eingeben: LIST"D:(Programmname)",100,300. Nun werden nur die Zeilen von 100 bis 300 auf Diskette gelistet.

LOAD (Lade)

Format : LOAD "Gerätespezifikation:Programmname"
Beispiele: LOAD "D:TEST1.BAS"
100 LOAD "C:"

Mit dem Befehl wird ein Programm in den Speicher des Computers geladen. Das Programm wird von dem Gerät geladen, das spezifiziert wurde ("C:" = Cassette / "D:" = Diskette.).

Mit LOAD können Daten oder mit LIST abgespeicherte Programme von Diskette oder Kassette geladen werden.

Bei der Verwendung des LOAD Befehls wird ein im Speicher stehendes Programm gelöscht (siehe MERGE Befehl auf S.22).

ATARINSIDE

LOCK (Blockieren)

Format : LOCK "Gerätespezifikation:Programmname"
Beispiel: LOCK "D:TEST1.BAS"

Mit dem LOCK-Befehl wird das Programm mit dem eingegebenen Namen auf der Diskette blockiert. Sie können es nun nicht mehr mit einem Programm gleichen Namens überschreiben. Auch ein Löschen oder umbenennen ist nicht mehr möglich. Aufheben geschieht mit dem Befehl UNLOCK (S.25). Im DOS- Menü erscheinen blockierte Programme mit einem Stern ( * ).

MERGE (Mischen)

Format : MERGE "Gerätespezifikation:Programmname"
Beispiel: MERGE "D:TEST1.BAS"

Mit diesem Befehl wird ein bereits im Speicher des Computers vorhandenes Programm mit dem aufgerufenen Programm gemischt. Dabei werden gleiche Zeilennummern überschrieben.

Beispielprogramm:
```
100 REM Dieses Programm
120 REM soll dann
130 REM vermischt
140 REM sein
   LIST "D:TEST1.BAS"  (RETURN-Taste)
   NEW (RETURN-Taste)
110 REM wird nun
120 REM mit dem ersten
   MERGE "D:TEST1.BAS"  (RETURN-Taste)
   LIST (RETURN-Taste)
100 REM Dieses Programm
110 REM wird nun
120 REM mit dem ersten
130 REM vermischt
140 REM sein
```

NAME...TO (umbenennen von Programmen, Dateien)
(Nur mit Erweiterungsdiskette.)

Format : NAME "Gerätespezifikation:alter Programmname" TO "neuer Programmname"
Beispiel: NAME "D:TEST1.BAS" TO "PROGRAMM.BAS"

Im Beispiel wird der Programmname TEST1.BAS in den Programmnamen PROGRAMM.BAS umgewandelt.



NEW (Neu)

Format : NEW
Beispiele: NEW
100 IF CODE <> 642 THEN NEW

Mit NEW wird das im Speicher stehende Programm gelöscht. Alle Variablen werden auf Null gesetzt und alle Strings werden gelöscht. Nur der Wert des Programmierwortes TIME$ bleibt erhalten.

RENUM (Neu numerieren)
(Nur mit Erweiterungsdiskette)

Format : RENUM l,m,n
Beispiel: RENUM 10,100,10
l = die erste Zahl der neuen Numerierung.
m = die Zeilennummer, ab der die Neunumerierung beginnen soll.
n = der Abstand der neuen Zeilennummern zueinander.

Ohne Zusatz beginnt die Numerierung bei der Zeile 10 und steigt um jeweils 10 auf.

Durch den Befehl RENUM werden auch alle GOTO, GOSUB, THEN, ON...GOTO, ON...GOSUB und ERROR Befehle in die neuen Zeilennummern verzweigt.

ANMERKUNG: Bei Zusatz von l, m oder n können natürlich nur Ziffern genommen werden, die auch bestehen. So wird z.B. der Befehl RENUM 15,30 nicht ausgeführt, wenn das Programm nur aus den drei Zeilen 10,20 und 30 besteht. Die höchste Zeilennummer ist 63999.
Beispiele:
RENUM 10,100 Die Zeile 100 wird zur Zeile 10.
Die Zeilen steigen um jeweils 10 an.
RENUM 800,900,20 Die Neunumerierung wird von Zeile
900 bis zum Endes des Programms durchgeführt.
Dabei ist die Zeile 900 nun 800. Die Zeilen
liegen jeweils um 20 auseinander.
RENUM 300,140,20:

| Vorher | Nachher |
|---|---|
| 100 | 100 |
| 110 | 110 |
| 120 | 120 |
| 130 | 130 |
| 140 | 300 |
| 150 | 320 |
| 160 | 340 |
| 170 | 360 |

ATARINSIDE

RUN (Start)

Format : RUN "Gerätespezifikation:Programmname"(optional Startzeilennummer)
Beispiele: RUN "D:PROGRAMM.BAS"
RUN 120
100 RUN "D:PROGRAMM.BAS"
150 RUN 1000

Wenn der Befehl RUN ohne Zusatz eingegeben wird, beginnt das im Speicher des Computers vorhandene Programm ab der niedrigsten Zeilennummer zu "laufen". Gleichzeitig werden alle Variablen auf Null gesetzt und vorhandene alphanumerische Variable (STRINGS) gelöscht.

Der Befehl kann auch während eines Programms gegeben werden. Er wird dann benutzt um ein neues Programm einzuladen. Der Zusatz einer Zeilennummer hinter dem Befehl ist in diesem Fall unzulässig. Durch RUN "D:Programmname" wird das im Speicherbereich stehende Programm gelöscht. Das aufgerufene Programm muß mit SAVE abgespeichert worden sein.

Mit RUN 1000 startet das Programm ab der Zeile 1000. Alle Variablen- und Stringwerte bleiben erhalten.

SAVE (Speichern)

Format : SAVE "Gerätespezifikation:Programmname"
Beispiele: SAVE "D:SPIEL.BAS"
SAVE "C:"

Mit SAVE wird das im Speicher des Computers stehende Programm auf Diskette oder Kassette abgespeichert. Bei Diskette ist die Eingabe eines Programmnamens erforderlich. Wie dieser Programmname sich aufbaut, lesen Sie bitte in Ihrem Handbuch zum Disketten Operations System nach.

ANMERKUNG: Ein Programm mit dem Namen AUTORUN.AMB wird beim Einschalten des Computers automatisch von Diskette geladen.

SAVE...LOCK (Speichere und blockiere)

Format : SAVE "Gerätespezifikation:Programmname" LOCK
Beispiele: SAVE "D:PROGRAMM.BAS" LOCK
SAVE "C:" LOCK

Mit diesem Befehl wird ein Softwareschutz erreicht. So abgespeicherte Programme können nicht mehr gelistet, verändert oder gemischt werden. Das Programm ist somit vor fremdem Zugriff gesichert. Aber denken Sie bitte daran: auch Sie haben keine Möglichkeit mehr, daß Programm zu listen oder zu verändern. Sie sollten also nur getestete Programme so abspeichern.



TROFF (Anzeigen der jeweiligen Zeilennummer - Aus).
(Nur in Verbindung mit Erweiterungsdiskette).

Format : TROFF
Beispiel: 770 TROFF

Dieser Befehl schaltet die nachfolgend aufgeführte Funktion TRON aus. Der Befehl TROFF kann direkt oder innerhalb eines Programms gegeben werden.

TRON (Anzeigen der jeweiligen Zeilennummer - An).
(Nur in Verbindung mit Erweiterungsdiskette).

Format : TRON
Beispiele: TRON
500 TRON

Mit diesem Befehl, (TRON steht für Trace ON), wird bei der Programmausführung die Zeilennummer auf dem Bildschirm angezeigt, die gerade abgearbeitet wird. Damit wird das Testen eines Programms wesentlich erleichtert. Der Befehl kann direkt oder innerhalb eines Programms gegeben werden.

UNLOCK (Blockierung aufheben)

Format :UNLOCK "Gerätespezifikation:Programmname"
Beispiel:UNLOCK "D:PROGRAMM.BAS"

Mit diesem Befehl wird der auf Seite 22 beschriebene Befehl LOCK wieder aufgehoben. Das genannte Programm kann wieder überschrieben, gelöscht oder umbenannt werden. Den Schutz der mit SAVE...LOCK abgespeicherten Programme können Sie damit nicht aufheben.

VERIFY (Überprüfen)
(Nur in Verbindung mit Erweiterungsdiskette)

Format : VERIFY "Gerätespezifikation:Programmname"
Beispiele: VERIFY "D:PROGRAMM.BAS"
VERIFY "C:"

Dieser Befehl dient zum Vergleich des abgespeicherten Programms mit dem noch im Speicher des Computers befindlichen gleichem Programm. Wird bei der Prüfung festgestellt, daß Abweichungen vorhanden sind, erscheint eine Fehlermeldung.

ATARINSIDE

# 4 PROGRAMM ANWEISUNGEN

AFTER (nach...)

Format : AFTER (Zeit in 1/50 Sekunden) GOTO (Zeilennummer)
Beispiel: AFTER (266) GOTO 220

Wenn die Anweisung AFTER im Programm auftritt, wird eine interne Uhr auf null gesetzt und beginnt in 1/50 Sekunden bis zu der in Klammern gesetzten Zahl zu zählen. Danach erfolgt die Ausführung des Befehls in "Zeilennummer". Eine Verzögerung bis 24 Stunden ist möglich.

Mit RUN, STOP oder END wird die interne Uhr wieder auf null gesetzt.
Beispielprogramm:

```
10 CLS:AFTER (250) GOTO 70
20 PRINT "SIE HABEN 5 SEKUNDEN ZEIT":PRINT "UM EINE TASTE
   ZU DRUECKEN."
30 IF INKEY$="" THEN 30
40 PRINT "DANKE SCHOEN"
50 CLEAR STACK
60 END
70 PRINT "DIE ZEIT IST ABGELAUFEN";
80 RESUME 50
```

CLEAR (Zurücksetzen)

Format : CLEAR
Beispiele: CLEAR
100 CLEAR

Mit CLEAR werden alle Variablen und alphanumerischen Varaiblen (STRINGS) auf Null gesetzt bzw. gelöscht. Bei der Verwendung von Variablenfeldern muß eine neue Dimensionierung erfolgen, bevor das Programm weiter abgearbeitet werden kann.

CLEAR STACK (Zurücksetzen der Zähler)

Format : CLEAR STACK
Beispiel: 100 CLEAR STACK

Mit dieser Anweisung werden alle Zeitfunktionen auf Null gesetzt. Die Hauptverwendung liegt in der Verbindung mit der AFTER Anweisung.



CLOSE (Schließen)

Format :CLOSE # Datenkanal
Beispiel:CLOSE #1

Mit der CLOSE-Anweisung sollte nach einer Übertragung von Daten der mit der OPEN-Anweisung (S.35) geöffnete Kanal wieder geschlossen werden.

COMMON (Übernehmen)

Formate : COMMON Variablen Name (,Variablen Name)
COMMON ALL
Beispiele: 110 COMMON I,J,A$
100 COMMON ALL

Mit dieser Anweisung ist es möglich, den Wert von Variablen festzuhalten und in ein neues Programm zu übernehmen.
Beispielprogramm:

```
100 COMMON X
110 X=5
120 RUN "D:TEST.BAS"
```

Ohne die COMMON-Anweisung würde X beim Laden des Programms TEST.BAS wieder auf Null gesetzt werden. So aber wird der Wert X (hier 5) der Variablen X im Programm TEST.BAS zugewiesen. Mit COMMON ALL werden alle Variablen übernommen.

Damit ist es Ihnen möglich, bei mehreren miteinander arbeitenden Programmen bestimmte Werte zu übertragen. Wenn Sie zum Beispiel Programme erstellen, bei denen jeweils das Datum ausgedruckt wird, so braucht dieses nur einmal eingegeben zu werden.

DEF (Definieren)
(Nur in Verbindung mit Erweiterungsdiskette.)

Format : DEF Name der Funktion (Variable,Variable) = Definieren der Funktion
Beispiel: 150 DEF MULT(J,K) = J*K

Mit der DEF-Funktion wird einer Variablen ein bestimmter Wert zugewiesen. Geben Sie beispielsweise ein: DEF A(X) = X^2, so gibt A(X) den Wert des Quadrates aus X wieder. Diese Anweisung wird nachfolgend noch an Beispielen genauer erklärt. Zunächst einmal grundsätzliche Erläuterungen. Der Befehl soll zweckmäßigerweise am Anfang eines Programms stehen. Er darf nur allein in einer Zeile stehen. Das Definieren von STRINGS ist ebenfalls möglich. So ist z.B. DEF S$(A$,B$) = A$+B$ zulässig.

ATARINSIDE

Beispielprogramme:

```
5 REM DEFINIEREN ZUR MITTELWERTBERECHNUNG EINER FUNKTION
10 DEF FUN(X,Y) = (X+Y)/2
20 PRINT FUN(25,35)
30 END

RUN
30

100 REM LESEN DER PADDLE POSITION
110 DEF PADDLE(X) = PEEK(642+X)
120 PRINT PADDLE(0)
130 GOTO 120

100 REM ABFRAGE DES FEUERKNOPFES
110 DEF STRIG(X) = PEEK(644+X)
120 IF STRIG(0) THEN 120 ELSE PRINT "PENG ";
130 GOTO 120
```

ANMERKUNG: DEF kann nicht im Direktmodus eingegeben werden.

## DIM (Dimensionieren)

Formate : DIM Name der numerischen Variable (benötigte Anzahl)
DIM Name der alphanumerischen Variablen (benötigter Speicherplatz)
Beispiele: DIM A(10),A$(100)

Mit DIM wird der dimensionierten Variablen ein bestimmter Speicherplatz freigehalten. Wenn dieser Speicherplatz überschritten wird, erfolgt eine Fehlermeldung. Mit der Dimensionierung numerischer Variablen können Sie ein- oder mehrdimensionale Felder bestimmen.

Beispiel:

```
10 OPTION BASE 1
20 DIM PUNKTE (26)
30 PUNKTE(1)=55
40 PUNKTE(2)=46
50 PUNKTE(3)=52
60 FOR I=1 TO 3: PRINT PUNKTE(I): NEXT I
```

Der in Zeile 10 verwendete Befehl bewirkt, daß das erste Element mit 1 indiziert wird (siehe auch S.15 ff / FELDER).

Im ATARI Microsoft BASIC II ist eine Dimensionierung von 255 Feldern erlaubt. Mehrdimesionale Felder (Matrizen) werden im nächsten Beispiel erklärt:

```
110 X=10:Y=10:Z=10
120 DIM KISTEN(X,Y,Z)
130 KISTEN (3,4,1)=123.45
140 KISTEN (3,5,1)=12.88
```

So können Sie alle "KISTEN" mit einem Wert füllen, der Ihnen bei Bedarf zur Verfügung steht.(PRINT KISTEN(3,5,1))

```
10 DIM GRUPPE1(10)
20 FOR I=0 TO 10:GRUPPE1(I)=I:PRINT GRUPPE1(I):NEXT
30 END
```



## END (Ende)

Format : END
Beispiel: 20000 END

Mit END wird die Programmausführung beendet. Es sollte deshalb immer in der letzten Zeile stehen. Beim Microsoft BASIC II ist das END nicht unbedingt erforderlich.

## ERROR (Fehler)

Format : ERROR Fehlernummer
Beispiel: 100 ERROR 162

Diese Anweisung dient zum Testen von Programmen. Wenn Sie ERROR gefolgt von einer Fehlernummer eingeben, sehen Sie, wie sich das Programm im Fall eines solchen Fehlers verhält. Welcher Fehlertyp welche Nummer hat, können Sie aus dem ANHANG O ersehen.

## FOR...TO...STEP / NEXT

Format : FOR Variable = Anfangswert TO Endwert STEP Schrittweite
Beispiele: 10 FOR X=1 TO 500 STEP 3
150 FOR I=400 TO 500
200 FOR ZEIT=100 TO 1 STEP -1

Eine sogenannte FOR / NEXT-Schleife wird so oft durchlaufen, bis die Variable den Endwert erreicht hat. Dabei erhöht oder erniedrigt sie sich nach jedem Durchlauf um den durch STEP angegebenen Wert. Ist kein STEP angefügt wird sie immer um eins erhöht.

Beispielprogramme:

```
10 FOR X=20 TO 100
20 PRINT X,SQR(X)
30 NEXT

10 FOR I=1 TO 255 STEP 10
20 POKE 710,I
30 NEXT
```

## GET (Nehme !)

Format : GET# Datenkanal,AT(Sektor, Byte);Variablenname(n)
Beispiele: 200 GET #1,X
240 GET #3,AT(8,2);J,K,L

GET liest ein Byte, (Wert zwischen 0 und 255), von dem geöffneten Datenkanal und weist der Variablen den Wert zu. Bevor Sie das nachfolgende Beispielprogramm eingeben, muß die Datei mit dem PUT Befehl (S.42) auf Diskette angelegt sein.

ATARINSIDE

Beispielprogramm:
```
110 OPEN #1,"D:TEST" INPUT
120 GET #1,A,B,C
130 CLOSE #1
140 PRINT A,B,C
```

Amnerkung: GET darf nicht im Direktmodus verwendet werden.

GOSUB / RETURN (Geh' zur Unterroutine / Kehre zurück)

Format : GOSUB Zeilennummer
Beispiel: 330 GOSUB 150

Mit GOSUB können ständig wiederverwendete Unterroutinen angesprungen werden. (z.B. Zeitschleifen, Tastaturabfragen o.ä.). Mit RETURN wird das Ende dieser Unterroutine gekennzeichnet. Das Programm wird an der Stelle weiterverarbeitet, an der das GOSUB steht.

GOTO (Geh' zur Zeile)

Format : GOTO Zeilennummer
Beispiel: 10 GOTO 110

Mit GOTO wird dem Computer mitgeteilt, mit welcher Zeile er das Programm fortsetzen soll. Es wird auch oft im Zusammenhang mit Vergleichen verwendet. (IF A=B THEN GOTO 1000).
Beispielprogramm:
```
10 PRINT "ENDLOSSCHLEIFE " ;
20 GOTO 10
```

Dieses Programm kann nur durch Drücken der BREAK-Taste angehalten werden.

IF...THEN (Wenn... dann)

Format : IF logischer Ausdruck THEN gehe zu Zeile oder Anweisung
Beispiele: 10 IF A=B THEN 290
20 IF A<31 AND A>128 THEN PRINT "Neuer VERSUCH"

Immer wenn ein logischer Ausdruck ungleich 0 ist, wird das in der Zeile stehende Statement ausgeführt. Als Besonderheit ist zu bemerken, daß das GOTO hier entfallen kann.

Trifft ein Vergleich nicht zu, wird die nächste Programmzeile abgearbeitet.
Beispielprogramm:
```
100 INPUT A,B
120 IF A > B THEN 200
130 PRINT "B ist groesser"
140 STOP
200 PRINT "A ist groesser"
450 END
```



IF...THEN...ELSE (Wenn...dann...sonst)

Format : IF logischer Ausdruck THEN gehe zu Zeilennummer oder Anweisung
ELSE gehe zu Zeilennummer oder Anweisung
Beispiel: 250 IF R<Y THEN 450 ELSE 200

In der Funktion genau wie die IF...THEN-Anweisung. Nur wird hier das Programm mit der hinter ELSE stehenden Anweisung weitergeführt, wenn der logische Ausdruck nicht zutrifft.
Beispielprogramm:
```
100 INPUT A,B
110 IF A>B THEN 200 ELSE PRINT "B IST GROESSER"
120 STOP
200 PRINT "A IST GROESSER":END
```

INPUT (Eingabe)

Format : INPUT # Datenkanal ("Text");Variablenname
Beispiele: 100 INPUT "Geben Sie Ihren Namen ein";A$
110 INPUT "Kontonummer,Kontoinhaber";KTO,KI$
100 INPUT #1,A$

Mit der INPUT-Anweisung wird zwischen dem Anwender und dem Computer ein scheinbarer Dialog geführt. In den ersten beiden Beispielen wird eine Eingabe erwartet. Diese muß mit der RETURN-Taste abgeschlossen werden. Das Programm wartet bis die Eingabe erfolgt ist. Die Aufforderung zur Eingabe wird durch ein Fragezeichen angezeigt. Die INPUT-Anweisung kann nicht im Direktmodus gegeben werden.

INPUT...AT (Eingabe..von)

Format : INPUT # Datenkanal,AT(Sektor,Byte) Variablenname
Beispiel: INPUT #1,AT(9,7)X

Wenn der Datenkanal zum Diskettenlaufwerk geöffnet wurde, dann können der Variablen bestimmte Werte zugewiesen werden. Dazu werden der Sektor und das Byte angegeben, von welchen der Wert gelesen werden soll. Auf diese Art ist zum Beispiel auch das Lesen eines Wertes einer bestimmten Bildschirmposition möglich (INPUT #6,AT(x,y)X).

LET (Lasse)

Format : LET Variablenname = Wert oder Variablenname
Beispiele: 100 LET A=55
110 LET B=A
120 LET A=B*4/(4.22^4)
130 D=123

LET weist einer Variablen einen bestimmten Wert zu. Das Statement LET kann, wie im obigen Beispiel in Zeile 130, weggelassen werden.

ATARINSIDE

LINE INPUT (Zeilen Eingabe)

Format : LINE INPUT #Datenkanal ("Text");Variablenname
Beispiel: 190 LINE INPUT AB$

Im Prinzip mit der gleichen Funktion wie die INPUT-Anweisung erlaubt der LINE INPUT die Verwendung von Komma, Semikolon und Anführungsstrichen. Die Eingabe wird nicht mit einem Fragezeichen angekündigt.

LINE INPUT...AT (Zeile Eingabe...von)

Format : LINE INPUT #Datenkanal,AT(Sektor,Byte) Variablenname
Beispiel: 100 LINE INPUT #1,AT(9,7)X

Identisch mit der Anweisung INPUT...AT. Nur das auch hier Komma, Semikolon und Anführungszeichen verwendet werden können.

MOVE (Bewege !)

Format : MOVE von Speicheradresse,zur Speicheradresse,Anzahl Bytes
Beispiel: 20 MOVE 57344,16384,1024
10 MOVE ADR(A$),1536,20

Mit der MOVE-Anweisung wird eine bestimmte Anzahl von Bytes von einem Speicherbereich in einen anderen kopiert. Der Vorteil liegt in der hohen Geschwindigkeit. Beim ersten Beispiel wird zum Beispiel der im ROM-Bereich liegende Zeichensatz (Startadresse des ersten Zeichens ist 57344) in den RAM-Bereich ab Adresse 16384 kopiert. Dabei werden 1024 Werte kopiert; (pro Zeichen 8 Byte). Eine weitere Anwendung liegt im schnellen Bewegen von Player-Missile Objekten.

NEXT (Nächste)

Format : NEXT (Variablenname)
Beispiele: 30 NEXT I,J
40 NEXT ZEIT
50 NEXT

Mit NEXT wird die FOR NEXT-Schleife so oft durchlaufen, bis die Variable den vorbestimmten Wert erreicht hat. (Siehe auch S.29). Ohne Zusatz eines Variablennamens wird die innerste FOR NEXT-Schleife angesprungen. Die Variablenangabe ist also nur bei verschachtelten Schleifen erforderlich.



Beispielprogramme:

```
100 FOR I=10 TO 100 STEP 10
110 PRINT I
120 NEXT
130 END

100 FOR X=1 TO 20
110 FOR Y=1 TO 20
120 FOR Z=1 TO 20
130 NEXT Z,Y,X
```

NOTE (Zeichen)
(Nur in Verbindung mit Erweiterungsdiskette)

Format : NOTE #Datenkanal,Variablenname,Variablenname
Beispiel: 120 NOTE #4,I,J

Durch NOTE wird der gerade benutzte Diskettensektor als Zahl in die Variable mit dem ersten Namen geschrieben (im Beispiel in die Variable I). Der zweiten Variablen wird der Wert des nächsten Bytes zugewiesen. Dadurch läßt sich feststellen, an welcher Stelle auf der Diskette das nächste Byte gelesen oder geschrieben wird.

ON ERROR (bei Fehler)

Format : ON ERROR GOTO Zeilennummer
Beispiel: 10 ON ERROR 550

Normalerweise wird die Programmausführung bei einem autretenden Fehler unterbrochen und eine Fehlermitteilung erscheint auf dem Bildschirm. Wenn in eine der ersten Zeilen ein ON ERROR gefolgt von einer Zeilennummer geschrieben wird, verzweigt sich das Programm im Fehlerfall zu dieser Zeilennummer. Dies wird in dem Beispielprogramm deutlich.

Diese Anweisung muß in das Programm aufgenommen werden, bevor der erste Fehler auftauchen kann.

In der angesprungenen Zeile können dann verschiedene Befehle gegeben werden, so z.B. Mitteilungen, Warteschleifen, Tonsignale o.ä. Soll die Programmausführung danach fortgesetzt werden, ist dies nur mit der Anweisung RESUME möglich .

Wird das Programm mit END oder STOP beendet, wird die ON ERROR Anweisung aufgehoben. Das gilt auch bei Verwendung des Befehles RUN.

ATARINSIDE

Beispielprogramm:
```
10 ON ERROR 1000
20 PRINT #3,"ZEILE"
30 STOP
1000 PRINT "DATENKANAL NICHT GEOEFFNET"
1010 STOP
1020 RESUME
```

Die ON ERROR-Anweisung kann während des Programms gelöscht werden. Geben Sie dazu ein: Zeilennummer ON ERROR 0.

## ON...GOSUB / RETURN

Format : ON Variable GOSUB Zeilennummer,Zeilennummer,Zeilenummer
Beispiel: ON X GOSUB 400,500,700

Diese Anweisung verzweigt das Programm in Abhängigkeit von einer Variablen (im Beispiel X). Ist der Wert der Variablen 1, wird die erste Zeilennummer angesprungen. Ist der Wert 2, die zweite Zeilenummer und so weiter.
Beispielprogramm:
```
100 INPUT "GEBEN SIE EINE ZAHL EIN (1-4)";X
105 IF X<1 OR X>4 THEN 160
110 ON X GOSUB 130,140,150,170
120 GOTO 100
130 PRINT "SIE WAEHLTEN 1":RETURN
140 PRINT "SIE WAEHLTEN 2":RETURN
150 PRINT "SIE WAEHLTEN 3":RETURN
160 PRINT "SIE WAEHLTEN FALSCH":END
170 PRINT "PROGRAMM ENDE":END
```

## ON...GOTO

Format : ON Variable GOTO Zeilennummer,Zeilennummer,Zeilennummer
Beispiel: 400 ON X GOTO 500,1000,1500

Im Prinzip die gleiche Anweisung wie bei ON...GOSUB, nur daß hier ein GOTO verwendet wird und somit kein RETURN erforderlich ist. Auch hier wird, wenn die Variable den Wert 1 hat, die erste Zeilennummer, beim Wert von 2 die zweite Zeilennummer usw. angesprungen.



## OPEN (öffne)

Format : OPEN #Datenkanal,"Gerätespezifikation:Programmname" Funktion
Beispiele: 130 OPEN#4,"K:" INPUT
140 OPEN #3, "P:" OUTPUT
150 OPEN #3, "D:PROGRAMM.BAS" INPUT
160 OPEN #1, "D:DATEI.ADR" UPDATE
130 OPEN #5, "D:PROGRAMM.BAS" APPEND

Mit der Anweisung OPEN wird ein Datenkanal zu einem bestimmten Eingabe-/ Ausgabe-Peripheriegerät geöffnet.

Es stehen insgesamt 8 dieser Datenkanäle zur Verfügung. Davon ist jedoch der Kanal 0 fest für die Bildschirm Aus- und Eingabe reserviert. Er kann vom Anwender nicht benutzt werden.

Der Kanal #6 ist ebenfalls fest belegt. Mit ihm werden in den Grafikmodi die Ein- und Ausgaben abgewickelt. Beispiel:
```
10 GRAPHICS 2
20 PRINT #6, AT(5,5);"Bildschirmtest"
30 GOTO 30
```
Nachfolgend noch einmal die möglichen Gerätespezifikationen:
P: steht für PRINTER (Drucker). Nur OUTPUT möglich.
E: steht für EDITOR .
K: steht für KEYBORD (Tastatur). Nur INPUT möglich.
C: steht für CASSETTE (Kassettenrekorder).
S: steht für SCREEN (Bildschirm).
R: steht für RS 232-C. Dies ist die Bezeichnung für ein bestimmtes Interface.
D: steht für DISK (Diskette). Bei Aufrufen dieses Peripheriegerätes muß immer noch ein Programmname angefügt werden. Er soll immer mit einem Buchstaben beginnen und besteht aus maximal acht Zeichen und einer dreistelligen Zusatzkennung.

FUNKTIONEN:
INPUT = Eingabe vom Gerät
OUTPUT = Ausgabe ans Gerät
UPDATE = Eingabe und Ausgabe möglich
APPEND = Anfügen an eine bereits bestehende Datei

Der Vorteil dieser Datenkanäle liegt unter anderem darin, daß sie, wenn sie einmal geöffnet worden sind, ganz einfach angesprochen werden können, z.B. mit den Anweisungen PRINT#2, INPUT#1, NOTE#5, STATUS#2, GET#4 und PUT#4. Ein einfaches Beispiel soll dies veranschaulichen:
```
100 OPEN #3,"P:"OUTPUT
110 PRINT #3,"Dies ist ein Probeausdruck"
120 CLOSE #3
```

```
100 OPEN #1,"K:"INPUT
110 OPEN #2,"E:"OUTPUT
120 GET #1,A
130 IF A>31 AND A<123 THEN PUT #2,A
140 GOTO 120
```

Das zweite Programm müssen Sie mit BREAK stoppen.

ATARINSIDE

OPTION BASE

```
Formate : OPTION BASE 0
          OPTION BASE 1
Beispiele: 150 OPTION BASE 1
           200 DIM Z(25,25,25)
```

Die Anweisung OPTION BASE bestimmt, ob die Indizierung eines Feldes bei 1 oder 0 beginnt. Dazu muß die Anweisung am Anfang eines Programms stehen. Wird die Anweisung nicht verwendet, beginnt die Indizierung bei 0, d.h. die erste Variable ist z.B. A(0,0) statt A(1,1).
Beispielprogramm:

```
100 OPTION BASE 1
110 DIM MATRIX (15,15)
120 READ MATRIX(1,1),MATRIX(2,2),MATRIX(15,15)
130 DATA 32,33,34
140 PRINT MATRIX(1,1),MATRIX(2,2),MATRIX(15,15)
150 END
```

OPTION CHR1, OPTION CHR2, OPTION CHR0

```
Formate : OPTION CHR1
          OPTION CHR2
          OPTION CHR0
Beispiele: 110 OPTION CHR1
           100 OPTION CHR2
           200 OPTION CHR0
```

Mit der Anweisung OPTION CHR1 werden im Speicherbereich des Computers 1024 Byte freigehalten. Diese sind für einen zusätzlichen Zeichensatz gedacht. Mit OPTION CHR2 werden nur 512 Byte freigehalten.

OPTION CHR0 löscht diese Reservierung wieder.

Mit dem MOVE-Befehl können Sie den Zeichensatz aus dem ROM-Bereich an die freigehaltene Stelle kopieren. Sie haben auch die Möglichkeit, einen völlig neuen Zeichensatz zu erstellen. Mit den Anweisungen VARPTR(CHR1) oder VARPTR(CHR2) ermitteln Sie die Adresse des reservierten Speicherplatzes. Diese Startadresse kann dann in die Speicherzelle CHBAS gePOKEt werden. (Siehe auch Anhang K).
Beispielprogramm:

```
300 CHBAS=&2F4
310 ADDR%=VARPTR(CHR1)
320 POKE CHBAS,((ADDR%/256)AND&FF)
```

Ein Grafikbefehl muß immer vor der Anweisung OPTION CHRn erfolgen. Dies ist erforderlich, weil erst nach der Erstellung des Grafikmodus die Berechnung des erforderlichen Speicherplatzes erfolgt.

Wie verschiedene Zeichensätze untereinander umgeschaltet werden, sehen Sie im Anhang C.



OPTION PLM1, OPTION PLM2, OPTION PLM0

```
Formate : OPTION PLM1
          OPTION PLM2
          OPTION PLM0
Beispiele: 100 OPTION PLM1
           200 OPTION PLM2
           400 OPTION PLM0
```

Im Prinzip handelt es sich um die gleichen Funktionen wie die Anweisungen OPTION CHRn. Nur wird hier der Speicherbereich für die Erstellung von PLAYER MISSILE-Objekten (Spielfiguren) freigehalten.

Mit OPTION PLM1 werden 1280 Byte reserviert (eine Zeile Auflösung). Bei OPTION PLM2 werden 640 Byte reserviert (zwei Zeilen Auflösung). Mit OPTION PLM0 wird der Speicherbereich wieder freigegeben.

Auch hier muß erst die Grafik-Anweisung und dann die OPTION PLMn-Anweisung erfolgen.

Bei der Benutzung von OPTION PLM1 oder / und OPTION PLM2 wird der Speicherbereich gelöscht und die PMBASE-Adresse wird gesetzt (siehe auch Tabelle E-2 in Anhang L). Die genaue Position des ersten reservierten Bytes wird durch die Anwendung von VARPTR(PLM1) oder VARPTR(PLM2) ermittelt.

In die Adresse Dezimal 53277 müssen Sie eine 3 POKEn um die Player Missile-Grafik anzusprechen. Weiterhin bestimmen Sie durch Ändern des Inhaltes der Adresse 559 die Auflösung des Players (POKE 559,62 = eine Zeile / POKE 559,46 = zwei Zeilen Auflösung). In Abschnitt 7 ist ein Beispielprogramm zur Erstellung der PM-Grafik abgedruckt.

OPTION RESERVE

```
Format : OPTION RESERVE Nr
Beispiel: OPTION RESERVE 24
```

Mit dieser Anweisung können Sie einen bestimmten Speicherbereich reservieren lassen. Im Beispiel wären es 24 Byte. Dies kann erforderlich sein, um Maschinensprache-Unterroutinen anzulegen o.ä.
Mit der Anweisung VARPTR(RESERVE) ermitteln Sie, wo das erste dieser 24 Bytes im Speicherbereich reserviert ist.

PRINT (Schreibe!)

```
Formate : PRINT "Text"
          ? "Text",Variablenname
          PRINT Variablenname,Variablenname,Variablenname,"Text"
Beispiele: 100 PRINT "Programm ";A$;" wird sortiert."
           500 ?#6;"FALSCH !"
```

Mit der Anweisung PRINT werden numerische und alphanumerische Variablen oder Konstanten auf dem Bildschirm ausgegeben. Ein PRINT allein schreibt eine Leerzeile. Statt des Ausschreibens des Wortes PRINT kann auch ein Fragezeichen ( ? ) eingegeben werden. Es übt die gleiche Funktion aus.

ATARI INSIDE

Beispielprogramm:
```
100 PRINT "ERSTE ZEILE "
110 PRINT
120 Z$="ZWEITE ZEILE"
130 ? Z$
140 ?:?"DRITTE ""ZEILE""
```

PRINT...AT

Formate: PRINT #Datenkanal,AT(S,B)x,y
PRINT #6,AT(x,y)"Text";Variablenname

Mit dieser Anweisung werden die Variablen oder Konstanten an eine bestimmte Stelle geschrieben.

Soll ein Wert auf eine Diskette geschrieben werden, wird mit S die Sektornummer und mit B die Bytenummer angegeben. Wenn der Datenkanal #6 (S:) angesprochen wird, repräsentieren S und B die Spalte und Zeile, ab der geschrieben wird (eröffnen des Datenkanals #6 geschieht automatisch bei Eingabe von GRAPHICS 1 oder GRAPHICS 2).
Beispielprogramme:
```
100 OPEN#3,"D:TEST.DAT" OUTPUT
110 X=5
120 ?#3,AT(7,1)"TEST";X
130 CLOSE#3
```

ANMERKUNG: Die Sektor- und Byte-Position muß vorher ermittelt werden. Siehe dazu Anweisung NOTE (S.33).

```
100 GRAPHICS 1
110 ?#6,AT(3,3)"GRAFIKTEST"
```

PRINT...SPC

Format : SPC(n)
Beispiel: 10 PRINT TAB(5);"xyz";SPC(7);"7 Leerstellen nach xyz"

Mit SPC (Space = Zwischenraum) bestimmen Sie die Anzahl der Leerstellen vom letzten geschriebenen bis zum ersten neuen Zeichen.

PRINT...TAB

Format : TAB(n)
Beispiel: 120 PRINT TAB(5);"Beginn an 5. Stelle"

Mit der Anweisung TAB springt der Cursor an die in Klammer angegebene Position innerhalb der Zeile. Dabei ist der linke Zeilenrand die Position TAB(0).
Beispielprogramme:
```
100 PRINT "Diese Zeile beginnt bei TAB(0)"
110 PRINT TAB(5);"Diese Zeile beginnt bei TAB(5)"
120 END
```



PRINT USING
(Nur in Verbindung mit Erweiterungsdiskette)

Mit dieser Anweisung haben Sie eine Vielzahl von Möglichkeiten die Ausgabe von Variablen zu beeinflussen. Da diese Funktionen sich am besten an Beispielen erläutern lassen, sind diese nachfolgend aufgeführt.

PRINT USING #

Jedes Nummernzeichen ( # ) steht für eine Stelle der zu erwartenden Ziffer, d.h. wenn Sie eine Ziffer mit maximal 7 Stellen erwarten, müssen Sie 7 Nummernzeichen setzen.
```
10 X=246
20 PRINT USING "###";X
```

Im Beispiel würde die Zahl 246 auf den Bildschirm geschrieben werden. Wenn die Variable X danach den Wert von 50 annimmt, wird diese 50 rechtsbündig unter die 246 geschrieben:
```
 246
  50
```

Dies trifft auf eventuell nachfolgende Ziffern ebenfalls zu. Sollten Sie nun der Variablen einen Wert zuweisen, der mehr Stellen hat, als Nummernzeichen vorhanden sind, wird diese Zahl mit vorangestelltem Prozentzeichen ( % ) gekennzeichnet.
```
100 X=99999
110 PRINT USING "###";X
RUN RETURN
%99999
```

PRINT USING.

Durch den Zusatz eines Dezimalpunktes ( . ) wird der Punkt bei der Ausgabe auch gesetzt. Die Ausgabe erfolgt wieder rechtsbündig. Sollte der Wert der Variablen keine Nachkommastellen haben, wird automatisch pro Nummernzeichen eine 0 angefügt:
```
10 X=2.468
20 PRINT USING "#.###";X
30 X=100
40 PRINT USING "###.###";X
50 X=2.468
60 PRINT USING "#.##";X
RUN RETURN
2.468
100.000
2.47
```

Im Beispiel können Sie auch erkennen, daß bei nur zwei Nummernzeichen hinter dem Punkt vor der Ausgabe automatisch aufgerundet wird. (Zeile 50 und 60).

ATARI INSIDE

PRINT USING,

Durch den Zusatz eines Kommas in der PRINT USING-Anweisung wird, von rechts beginnend, alle drei Stellen ein Komma eingefügt. Dies erleichtert das Lesen großer Zahlen.
Beispiel:

```
5 DEFDBL X
10 X=2933604.53
20 PRINT USING "########,##";X
RUN
2,933,604.53
```

PRINT USING **

Zwei Sterne an den ersten beiden Positionen füllen die Leerstellen vom Zeilenanfang bis zur Ziffer mit Sternen:

```
100 X=259
110 PRINT USING "*********.##";X
120 X=10.50
130 PRINT USING "*********.##";X
RUN
******259.00
*******10.50
```

PRINT USING $ oder DM

Durch den Zusatz des Dollarzeichens oder der Abkürzung DM wird der Zusatz links von der Zahl ausgegeben:

```
100 X=3.5968
110 PRINT USING "DM###.##";X
RUN
DM 3.60
```

PRINT USING $$

Durch das voranstellen zweier Dollarzeichen wird ein fließendes Dollarzeichen bewirkt, d.h. es steht immer links an der Zahl:

```
100 X=3.5968
110 PRINT USING "$$###.##";X
RUN
$3.60
```

PRINT USING **$

Wenn zwei Sterne und ein Dollarzeichen in den ersten drei Positionen stehen, werden die leeren Stellen mit Sternen aufgefüllt. Dann wird ein fließendes Dollarzeichen gesetzt:

```
100 X=53.29
110 PRINT USING"**$********.##";X
RUN
********$53.29
```



PRINT USING ^^^^

Vier an das Nummernzeichen angefügte Exponentenzeichen ( ^ ) wandeln eine Ziffer in eine Hochzahl um; (Exponentenfunktion E oder D):

```
100 X=500
110 PRINT USING "##^^^^";X
RUN
5E+02
```

PRINT USING +

Das Pluszeichen kann vor oder hinter der PRINT USING-Anweisung stehen. Es versieht die ausgegebene Zahl mit einem Pluszeichen bei positivem Wert und mit einem Minuszeichen bei negativem Wert:

```
100 A=999.95
110 PRINT USING "+####";A
RUN
+ 1000
```

PRINT USING -

Das Minuszeichen wird hinter die ausgegebene Ziffer gesetzt, sofern deren Wert negativ ist. Positive Zahlen werden ohne Zusatz ausgegeben:

```
100 A=-998
110 PRINT USING "----";A
RUN
998-
```

PRINT USING !

Mit dieser Anweisung wird das erste Zeichen einer alphanumerischen Konstanten oder Variablen ausgegeben.
Beispiel:

```
100 A$="A = ANTON"
110 PRINT USING "!";A$
RUN
A
```

PRINT USING %llll%

Durch Zusatz der Prozent- und Leerzeichen (l) können Sie bestimmte Teile alphanumerischer Konstanten oder Variablen ausgeben lassen. Dabei wird die Anzahl der auszugebenden Zeichen durch die Anzahl der Leerstellen plus der zwei Prozentzeichen bestimmt:

```
100 A$="Creative Computer Design"
110 PRINT USING "%llllllll%";A$
RUN
Creative
```

Probieren Sie die Möglichkeiten der PRINT USING-Anweisung alle aus. Sie werden noch eigene Ausgabearten entdecken, die Ihren speziellen Wünschen eher entsprechen, als diese allgemein gehaltenen Beispiele es können.

ATARINSIDE

PUT (Lege !)

Format : PUT #Datenkanal,(AT(Sektor,Byte))Variablenname
Beispiel: 100 PUT #6,AT(8,2)I,K,L

Die Anweisung PUT arbeitet in Verbindung mit der GET-Anweisung. PUT gibt einzelne Bytes mit einem Wert zwischen 0 und 255 an den geöffneten Datenkanal aus.

Das Beispielprogramm gibt 3 Zahlen an die Diskettendatei TEST aus. Diese können mit der GET-Anweisung wieder eingelesen werden.
Beispielprogramm:

```
100 OPEN #1,"D:TEST"OUTPUT
110 PUT #1,65,66,67
120 CLOSE #1
```

RANDOMIZE (zufällig)

Format : RANDOMIZE (Zahl)

Beispiele: 10 RANDOMIZE
10 RANDOMIZE 55

Normalerweise werden mit der arithmetischen Funktion RND immer die selben Pseudo-Zufallszahlen erzeugt. Das können Sie erkennen, wenn Sie folgendes Programm eingeben:

```
10 FOR J=1 TO 3:?:?
15 FOR I=1 TO 10
20 PRINT RND;
30 NEXT I,J
```

Sie werden feststellen, daß immer die gleichen Zahlen ausgegeben werden. Ändern Sie das Beispielprogramm durch Einfügen einer Zeile ab:

```
12 RANDOMIZE
```

Nun werden immer andere Zahlen ausgegeben. Um diesen Vorgang zu verstehen, stellen Sie sich den "Zufallszahlengenerator" als eine sehr lange Reihe von Zahlen vor. Ohne die RANDOMIZE-Anweisung werden die Zahlen jedesmal vom Anfang dieser Reihe beginnend ausgegeben. Mit RANDOMIZE beginnt die Ausgabe dieser Zahlen immer an einer anderen Stelle.

Diese Stelle können Sie bestimmen, indem Sie hinter die Anweisung eine Zahl setzen. Ändern Sie die Zeile 12 des Beispielprogramms ab:

```
12 RANDOMIZE 55
```



READ / DATA (Lesen / Daten)

Format : READ Variablenname1,Variablenname2,Variablenname3
Beispiele: 150 READ A,B,C
200 READ A$,B$

Mit der Anweisung READ werden bestimmten Variablen, numerischen oder alphanumerischen, bestimmte Werte zugewiesen. Diese Werte sind in einer sogenannten DATA-Anweisung (DATA-Zeile) festgelegt. Die Variablennamen müssen duch Kommata getrennt sein. Ohne RESTORE-Anweisung (siehe S.44) wird die DATA-Zeile immer weitergelesen. Nehmen Sie zum Beispiel an, Sie wollen den Variablen A,B und C die Werte 1,2 und 3 zuweisen. Dazu geben Sie ein:

```
10 READ A,B,C
100 DATA 1,2,3
```

Wenn Sie im weiteren Programmablauf noch einmal die READ-Anweisung geben, erscheint die Fehlermitteilung OUT OF DATA. Es stehen keine Daten (DATA) mehr zur Verfügung, weil die drei vorhandenen bereits zugewiesen wurden.

Sie können auch der selben Variablen ständig einen neuen Wert zuweisen:

```
100 FOR I=1 TO 3
110 READ A$,A
120 PRINT A$,A
130 NEXT
200 DATA FRED,50,JACK,20,JANE,200
300 PRINT "DATEI IST ZU ENDE !"
310 END
```

Wenn es erforderlich ist, können mehrere DATA-Zeilen erstellt werden. Das Auslesen erfolgt trotzdem der Reihenfolge entsprechend. Ausnahmen werden durch die RESTORE-Anweisung erreicht (S.44).

REM oder ! oder ' (Notiz, Anmerkung)

Format : REM
Beispiel: 10 REM hier koennen Sie Ihre Anmerkungen eingeben
20 LET R=25 !R steht fuer Radius
30 GOSUB 1000 'Unterroutine fuer Kreisberechnung
40 PRINT R: REM Ausgabe Kreisumfang

ATARINSIDE

```
Format : ! und '
Beispiel: 10 PRINT "Beispiel" !Ihre Anmerkung
          20 GOTO 10 ' es wurde ! und ' benutzt
```

Die REM-Anweisung gibt Ihnen die Möglichkeit, Ihre Programme mit Anmerkungen und Notizen zu versehen. Bei der Programmausführung werden diese nicht beachtet.

Das Ausrufungszeichen ( ! ) und das Apostroph ( ' ) können direkt hinter einem Befehl oder einer Anweisung eingegeben werden, während die Anweisung REM durch Doppelpunkt abgetrennt werden muß. Es ist auch erlaubt, Ihre Anmerkungen in inverser Schrift (ATARI-Taste) zu schreiben. Dadurch werden sie beim Listen eines Programmes noch deutlicher erkannt. Vergessen Sie aber bitte nicht, daß die Verwendung der REM-Anweisung Speicherplatz verbraucht.

## RESTORE

```
Format  : RESTORE (Zeilennummer)
Beispiele: 100 RESTORE 770
           300 RESTORE
```

Diese Anweisung arbeitet in Verbindung mit der READ / DATA-Anweisung. Durch ein RESTORE wird mit dem Lesen der ersten DATA-Zeile wieder von vorne begonnen.

Die Verwendung von RESTORE und einer Zeilennummer erlaubt das Dateneinlesen aus einer bestimmten DATA-Zeile. Dies soll mit einem Beispiel verdeutlicht werden:

```
10 REM READ DATA RESTORE DEMONSTRATION
20 DIM A(15)
30 FOR I=1 TO 10:READ A(I):PRINT A(I): NEXT
40 DATA 1,2,3,4,5
50 DATA 6,7,8,9,10
60 DATA 11,12,13,14,15
70 RESTORE 50: FOR N=1 TO 10:READ A(N):? A(N);: NEXT
```

## RESUME

```
Formate : RESUME Zeilennummer
          RESUME NEXT
          RESUME
Beispiele: 300 RESUME 55
           440 RESUME NEXT
           120 RESUME
```

Mit RESUME wird im Falle eines Fehlers die Programmausführung fortgesetzt. Voraussetzung ist hierbei jedoch, daß eine ON...ERROR Anweisung besteht (siehe S.33).

Mit RESUME (Zeilennummer) wird das Programm bei der angegebenen Zeilennummer fortgesetzt.

Mit RESUME NEXT wird das Programm mit der nach dem Fehler eingegebenen Anweisung fortgesetzt.

Mit RESUME ohne Zusatz wird erneut die Zeile angesprungen, in der der Fehler auftrat.



## RETURN (Zurück)

```
Format : RETURN
Beispiel: 550 RETURN
```

Diese Anweisung steht immer am Ende einer Unterroutine, die mit GOSUB oder ON...GOSUB aufgerufen wurde.

Beispielprogramm:

```
10 X=1: GOSUB 80: PRINT X
20 X=3: GOSUB 80: PRINT X
30 X=8.9: GOSUB 80: PRINT X
40 STOP
80 X=X^2: RETURN
```

## STACK (Stapel)

```
Format  : STACK
Beispiele: 100 PRINT STACK
           200 IF STACK = 0 THEN PRINT " STAPEL VOLL "
```

Die STACK-Funktion gibt an, wie viele Belegungen des Zeitzählers noch erlaubt bzw. möglich sind. Dies ist wichtig bei Verwendung der Anweisungen AFTER und SOUND. Es sind 20 Belegungen möglich, dann muß der Stappel zurückgesetzt werden (CLEAR STACK Seite 26).

## STOP (Stop)

```
Format :STOP
Beispiel: 100 STOP
```

Mit der STOP-Anweisung können Sie das Programm an jeder beliebigen Stelle anhalten. Es erscheint die Mitteilung BREAK IN und die Angabe der Zeilennummer. STOP dient zum Testen eines Programmes. Nach dem Abbruch ist es z.B. möglich, sich die Werte aller Variablen im Direktmodus ausgeben zu lassen. Das Programm wird nach der Eingabe von CONT fortgesetzt.

## VARPTR (Variablenpointer / Variablenstartadresse)

```
Formate : VARPTR(Variablenname)
          VARPTR(PLM1)
          VARPTR(PLM2)
          VARPTR(PLM0)
          VARPTR(CHR1)
          VARPTR(CHR2)
          VARPTR(RESERVE)
Beispiele: 110 A=VARPTR(A$)
           200 PRINT VARPTR(A$)+1
           100 J=VARPTR(TOTAL)
           320 T=VARPTR(CHR2)
           120 POKE VARPTR(RESERVE),&FE
```

ATARINSIDE

Mit dieser Anweisung läßt sich feststellen, an welcher Speicheradresse der für die jeweilige Funktion reservierte Speicherbereich beginnt.

Bei alphanumerischen Variablen (STRINGS) wird die Anzahl der Bytes angegeben. Die Startadresse ergibt sich dann so: VARPTR(A$)+1 ist das Low-Byte und VARPTR(A$)+2 ist das High-Byte. Bei der CPU 6502, die auch im ATARI verwendet wird, erfolgt die Eingabe von Werten in der Reihenfolge Low-Byte vor High-Byte. Bei der Abfrage der STRING-Startadresse werden diese Bytes bereits in der richtigen Reihenfolge bestimmt.

Folgende Programmierworte können bei der Anweisung VARPTR verwendet werden:

VARPTR(PLMn) enthält die Startadresse des für Player Missile-Grafik reservierten Speicherbereichs

VARPTR(CHRn) enthält die Startadresse des für einen zusätzlichen Zeichensatz reservierten Speicherbereichs

VARPTR(RESERVE) enthält die Startadresse des für Maschinensprache - Unterroutinen reservierten Speicherbereichs

Lesen Sie zu dieser Anweisung auch noch einmal die Erläuterungen bei den OPTION-Anweisungen (Seite 36 ff).

WAIT...AND (warte...und)

Format : WAIT Adresse,AND Maskenbyte Bedingungsbyte
Beispiel: 330 WAIT &D40B,&FF,110: REM Warten auf VBLANK

WAIT unterbricht die Programmausführung, bis der Inhalt der angegebenen Adresse undiert mit dem Maskenbyte dem Bedingungsbyte entspricht.

In unserem Beispiel wird gewartet, bis der VBLANK- (Vertical-Blank) Zähler einen bestimmten Wert erreicht hat (hier 110). Dann wird die Programmausführung fortgesetzt. Sie dürfen sich dieses "Warten" allerdings nicht als Verzögerung im üblichen Sinn vorstellen. Wenn Sie sich bereits mit den internen Zeitabläufen eines Computers beschäftigt haben, wissen Sie, daß diese Zeitspanne einige hundertstel Sekunden kurz sein kann.

Beispiele für die Verwendung der Anweisung WAIT finden Sie im Anhang C (Bewegung verschiedener Zeichensätze) und im Anhang A (Fein-Verschiebung).



# PROGRAMM FUNKTIONEN 5

NUMERISCHE FUNKTIONEN

ABS
Format : ABS(arithmetischer Ausdruck)
Beispiel: 10 ? ABS(-7)

Mit der Funktion ABS wird das Vorzeichen eines arithmetischen Ausdruckes als positiv gesetzt. In unserem Beispiel würde 7 (positive 7) ausgegeben werden.

ATN
Format : ATN(arithmetischer Ausdruck)
Beispiel: 10 ? ATN(.66)

Die Funktion ATN (Arcustangens) berechnet den Wert des Arcustangens, der zwischen Pi/-2 und Pi/2 liegt. Dessen Funktionswert wird in Radiant angezeigt. Im Beispiel ist das Ergebnis: .583373 rad.

COS
Format : COS(arithmetischer Ausdruck)
Beispiel: 10 ? COS(.95)

Die Funktion COS berechnet den Cosinus für den in Klammer stehenden arithmetischen Ausdruck. Im Beispiel lautet das Ergebnis: .581683

EXP
Format : EXP(arithmetischer Ausdruck)
Beispiel: 10 ? EXP(3)

EXP ergibt die Exponentialfunktion des arithmetischen Ausdrucks. Das Ergebnis im Beispiel ist 20.0855.

INT
Format : INT(arithmetischer Ausdruck)
Beispiele: ? INT(5.3) ! Ergebnis ist 5
? INT(-7.6) ! Ergebnis ist -8

INT ergibt die nächstniedrigere ganze Zahl.

LOG
Format : LOG(arithmetischer Ausdruck)
Beispiel: 10 ? LOG(5)

Berechnet den natürlichen Logarithmus des in Klammern stehenden Ausdrucks. Dieser muß größer als 0 sein. Im Beispiel lautet das Ergebnis 1.60944.

RND
Format : RND oder RND(0) oder RND(ganze Zahl)
Beispiele: ? RND (Ausgabe einer Zufallszahl mit 6 Nachkommastellen)
RND(37) (Ausgabe einer ganzzahligen Zufallszahl zwischen 0 und 38)

Mit der Funktion RND wird eine Zufallszahl ermittelt. Bei RND und RND(0) liegt diese zwischen 0 und 1. Sonst erzeugt RND eine Zufallszahl größer als 0 und kleiner / gleich der in Klammern stehender Zahl.

SGN
Format : SGN(arithmetischer Ausdruck)
Beispiel: ? SNG(-34)

Mit SGN wird das Vorzeichen des in Klammern stehenden Ausdrucks ermittelt. Ist der Ausdruck positiv, hat SGN den Wert +1. Ist der Ausdruck negativ hat SGN den Wert -1. Bei 0 ist SGN 0.

SIN
Format : SIN(arithmetischer Ausdruck)
Beispiel: 10 ? SIN(1)

SIN berechnet den Sinus des in Klammern stehenden arithmetischen Ausdrucks (im Beispiel: .841471).

SQR
Format : SQR(arithmetischer Ausdruck)
Beispiel: 10 ?SQR(25)

Mit der Funktion SQR wird die Quadratwurzel des in Klammern stehenden Ausdrucks errechnet. Die Zahl muß größer oder gleich 0 sein. SQR(25) = 5.

TAN
Format : TAN(arithmetischer Ausdruck)
Beispiel: 100 ? TAN(.22)

TAN berechnet den Tangens des Ausdrucks. Im Beispiel wird .223619 ausgegeben.

## STRING-FUNKTIONEN

+ (Verkettungsoperation)
Format : String+String
Beispiel: 110 C$=A$+B$
Das +Symbol wird benutzt, um zwei Strings aneinanderzufügen:

```
100 A$="HAUS"
110 B$="DACH"
120 C$=A$+B$
130 ? C$
RUN
HAUSDACH
```

ASC
Format : ASC(alphanumerischer Ausdruck)
Beispiel: ? ASC("Auto")
Mit der Funktion ASC wird das erste Zeichen des alphanumerischen Ausdrucks in den ATASCII-Code umgewandelt. Im Beispiel entspricht der Buchstabe A der Zahl 65. Siehe auch Anhang K.

CHR$
Format : CHR$(ATASCII-Code)
Beispiele: 110 PRINT CHR$(123):REM PIK Symbol
120 PRINT CHR$(65) :REM Buchstabe A
Mit CHR$ wird eine Zahl zwischen 0 und 255 in das Zeichen umgewandelt, das der Zahl im ATASCII-Code entspricht. Welche Codes welchem Zeichen zugeordnet ist, können Sie aus dem Anhang K ersehen.



INKEY$ (INput KEYboard / Eingabe von Tastatur)
Format : INKEY$
Beispiel: 110 A$=INKEY$

Mit der Funktion INKEY$ wird überprüft, ob und welche Taste gedrückt wurde. Das Zeichen der letzten gedrückten Taste wird mit INKEY$ übergeben. Damit lassen sich z.B. Auswahlen treffen u.ä.. Die Leertaste wird von INKEY$ ignoriert.
Beispielprogramm:

```
100 A$=INKEY$
110 IF A$<>"" THEN PRINT "SIE DRUECKTEN EIN ";A$
120 GOTO 100
```

INSTR
Format : INSTR(n,A$,B$)
Beispiel: 100 A = INSTR(5,C$,B$)

Mit dieser Funktion wird ein kleiner String (z.B.B$) innerhalb eines grösseren (z.B.A$) gesucht. Gesucht wird ab der mit n angegebenen Position. Ohne Angabe von n wird 1 angenommen. Der Wert dieser Funktion sagt aus, wo der String gefunden wurde. Eine 0 bedeutet, daß der gesuchte String nicht vorhanden ist. Ein Beispiel:

```
10 A$="ZEILENABSTAND":B$="EILE"
20 A=INSTR(A$,B$)
30 ?A
```

Der String B$ steht hier ab der zweiten Stelle im A$. A ist also 2.

LEFT$
Format : LEFT$(alphanumerischer Ausdruck,n)
Beispiel: 100 A$="Zeilenabstand"
110 ? LEFT$(A$,5)
Es wird der linke Teil des Strings bis zum n. Zeichen ausgegeben, im Beispiel also das Wort 'Zeile'.

LEN
Format : LEN(alphanumerischer Ausdruck)
Beispiel: 100 A$="Kontoinhaber"
110 A=LEN(A$)
LEN gibt die Länge des Ausdrucks an. Im Beispiel ist A=12.

MID$
Format : MID$(alphanumerischer Ausdruck,m,n)
Beispiel: 100 A$="Name des Kontoinhabers"
110 ? MID$(A$,10,5)
Mit der MID$-Funktion wird ein bestimmter Teil eines String ausgewählt. Dabei gibt m an, ab wo der Ausdruck ausgegeben wird. n gibt die Anzahl der Zeichen an. Im Beispiel wird Konto ausgegeben (10. Stelle, 5 Zeichen).

RIGHT$
Format : RIGHT$(alphanumerischer Ausdruck,n)
Beispiel: 100 A$="MAUSEFALLE"
110 ? RIGHT$(A$,5)
RIGHT$ ist das Gegenstück zu LEFT$. Hier werden nun die letzten n Zeichen des Ausdrucks ausgegeben. Im Beispiel werden die letzten 5 Zeichen der Variablen A$ ausgegeben (FALLE).

ATARIINSIDE

SCRN$
Format : SCRN$(X,Y)
Beispiel: 10 ? SCRN$(5,5)

Mit dieser Funktion wird ein Zeichen vom Bildschirm gelesen. Die Position, von der es gelesen wird, ist durch X und Y angegeben. In den Grafikmodi, in denen keine Zeichen auf dem Bildschirm vorhanden sind, wird der ATASCII-Code des Farbregisters des Punktes auf dem Bildschirm gelesen. Beispielprogramme:

```
10 GRAPHICS 1: COLOR 1
20 PRINT#6,AT(5,5);"A": A$=SCRN$(5,5)
30 PRINT "ZEICHEN IST ";A$

10 GRAPHICS 7: COLOR 3
20 PLOT 5,5: A$=SCRN$(5,5)
30 PRINT "FARBREGISTER IST ";ASC(A$)
```

STR$
Format : STR$(arithmetischer Ausdruck)
Beispiel: 100 A=999.02
110 PRINT STR$(A)

Mit der STR$-Funktion wird eine Zahl in einen STRING umgewandelt. Dies wird am besten an einem Beispiel deutlich:

```
10 NUM1=-22.344:NUM2=43.2
20 A$=STR$(NUM1) + STR$(NUM2)
30 PRINT A$
```

Es wird -22.344 43.2 ausgegeben.

STRING$(n,A$)
(Nur in Verbindung mit Erweiterungsdiskette)
Format : STRING$(n,A$)
Beispiel: 100 A$=STRING$(20,"*")

Ausgegeben wird n mal die in A$ stehende Zeichenkette. Im Beispiel 20 mal das Zeichen *.

STRING$(m,n)
(Nur in Verbindung mit Erweiterungsdiskette)
Format : STRING$(m,n)
Beispiel: 10 ? STRING$(20,42)

Es wird m mal das dem ATASCII-Code n entsprechende Zeichen ausgegeben. Im Beispiel 20 mal das Zeichen mit dem ATASCII-Code 42 ( * ).



TIME$
Format : TIME$="Stunde:Minute:Sekunden"
Beispiel: 100 PRINT TIME$

TIME$ gibt die Zeit in Stunden, Minuten und Sekunden an. Die Differenz beträgt plus/minus 90 Sek. pro 24 Stunden.
Beispiele:

```
100 TIME$="22:55:05"
120 TIME$="05:30:09"
```

Nachdem die Variable TIME$ einmal gesetzt ist, kann sie während des Programmablaufs ständig abgefragt werden:

```
100 GRAPHICS 2
110 TIME$="11:59:02"
120 ?#6,AT(3,3)" DIGITAL UHR"
130 ?#6,AT(4,4)TIME$
140 GOTO 120
```

VAL
Format : VAL (alphanumerische Variable)
Beispiel: 100 B$="309"
110 A=VAL(B$)
120 ? A

Mit dieser Funktion werden Zahlen oder Ziffern einer alphanumerischen Variablen (STRING) in eine numerische Variable umgewandelt. Sind in einer STRING-Variablen auch Buchstaben vorhanden, wird dort die Umwandlung abgebrochen. Beispiel B$="123ABC". Nach der Umwandlung wird nur 123 ausgegeben. Ist B$="ABC123" dann erfolgt die Ausgabe der Ziffer 0 .

SPEZIELLE FUNKTIONEN

EOF (End Of File / Ende der Datei)
Format : EOF(n)
Beispiel: 120 IF EOF(4)=0 THEN 60

Mit EOF wird der in Benutzung befindliche Datenkanal (n) überprüft. Wenn das Ende der Datei erreicht ist, nimmt EOF einen Wert von 1 an. Sonst ist der Wert 0.

ERL (Error Line / Zeile mit Fehler)
Format : ERL
Beispiel: 100 ? ERL

ERL gibt die Zeilennummer an, in der der letzte Fehler auftrat.

ERR (Error / Fehler)
Format : ERR
Beispiele: 120 ? ERR
200 IF ERR = 135 THEN 3000

ERR gibt die Codenummer des zuletzt aufgetretenen Fehlers aus.

ATARINSIDE

FRE(0)
Format : FRE(0)
Beispiel: PRINT FRE(0)

Mit der FRE(0)-Funktion wird der noch verbleibende Speicherplatz in Byte festgestellt. Sie wird vorwiegend im Direktmodus angewandt. Wird FRE(0) in Programmen benutzt, so werden unbenutzte Stringvariablen gelöscht.

PEEK
Format : PEEK(Adresse)
Beispiele: 100 ? PEEK(1034)
120 ? PEEK(ADDR)

Mit PEEK wird der Wert (Inhalt) einer bestimmten Speicherzelle ausgelesen. Dieser kann dann einer Variablen zugewiesen werden (A=PEEK(&FF)). Die Ausgabe erfolgt in dezimaler Schreibweise. Es können auch Adressen im ROM-Bereich ausgelesen werden. Beispiele:
? PEEK(888) gibt den Inhalt der dezimalen Adresse 888 aus.

? PEEK(&FF) gibt den Inhalt der hexadezimalen Adresse FF aus.

POKE
Format : POKE Adresse,Byte
Beispiele: POKE 2598,255
100 POKE ADDR,&FF
110 POKE CHBAS,J

Mit POKE ist es möglich, in eine bestimmte Adresse des RAM-Bereichs einen bestimmten Wert zu schreiben. Dieser Wert darf 255 oder FF hexadezimal nicht überschreiten. Durch das Umschreiben des Inhalts (Wertes) einer Adresse können bestimmte Funktionen schneller ausgeführt werden. Geben Sie z.B. einmal ein:
10 FOR I=1 TO 255: POKE 710,I: NEXT

Lesen Sie hierzu auch die Anlage E.

STATUS
Formate : STATUS (Datenkanalnummer)
STATUS ("Gerätespezifikation:Programmname)
Beispiele: 100 A=STATUS(6)
100 A=STATUS("D:TEST.BAS")

Mit STATUS wird der benutzte Datenkanal auf seinen Status überprüft. Liegt ein Fehler vor, wird ein bestimmter Wert in die Variable STATUS geschrieben. Dieser Wert entspricht jeweils einer Statuskennung, die durch die nachfolgende Tabelle erklärt wird. Dabei erkennt man aufgetretene Fehler daran, daß das höchstwertige Bit 1 ist.



Tabelle 5.1 Die STATUS-Codes

| Hex | Dez | Bedeutung |
|---|---|---|
| 01 | 001 | Übertragung beendet. Keine Fehler |
| 03 | 003 | Ende der Datei (EOF) |
| 80 | 128 | BREAK-Taste wurde gedrückt |
| 81 | 129 | der gewählte Datenkanal ist bereits geöffnet |
| 82 | 130 | das gewählte Gerät existiert nicht |
| 83 | 131 | Datenkanal ist nur zum Schreiben geöffnet |
| 84 | 132 | unvollständige / falsche Anweisung |
| 85 | 133 | Datenkanal / Datei nicht geöffnet |
| 86 | 134 | falsche Datenkanal-Nummer |
| 87 | 135 | Datenkanal ist nur zum Lesen geöffnet |
| 88 | 136 | Versuch über das Dateiende zu lesen |
| 89 | 137 | Datei abgeschnitten |
| 8A | 138 | Gerät antwortet nach einiger Zeit nicht |
| 8B | 139 | Negatives / nicht erfolgtes Quittungssignal vom Gerät |
| 8C | 140 | Lesefehler auf seriellem Bus |
| 8D | 141 | Cursor befindet sich außerhalb des Bildschirms |
| 8E | 142 | Computer verarbeitet eintreffende Daten zu langsam |
| 8F | 143 | Prüfsummen-Fehler |
| 90 | 144 | Gerätefehler |
| 91 | 145 | Unzulässiger Bildschirm-Modus |
| 92 | 146 | dem Datenkanal wurde eine unbekannte Anweisung gegeben |
| 93 | 147 | für den gewählten Grafikmodus zu wenig Speicherplatz |
| A0 | 160 | nicht vorhandene Diskettenstation angesprochen |
| A1 | 161 | zu viele offene Diskettendateien |
| A2 | 162 | Diskette ist vollgeschrieben |
| A3 | 163 | grober Lese-/Schreibfehler |
| A4 | 164 | Dateinummern / Sektoren vertauscht |
| A5 | 165 | falscher Dateiname |
| A6 | 166 | unzulässiger POINT-Befehl |
| A7 | 167 | Datei blockiert (LOCK-Befehl) |
| A8 | 168 | unzulässiger Diskettenbefehl |
| A9 | 169 | Direktory dieser Diskette ist voll (64 Namen) |
| AA | 170 | gewählte Datei befindet sich nicht auf der Diskette |
| AB | 171 | unzulässiger POINT-Befehl |

TIME (Zeit)
Format : TIME
Beispiel: 200 PRINT TIME

Im Gegensatz zur alphanumerischen Variablen TIME$ gibt die Variable TIME die Zeit nicht in Stunden, Minuten und Sekunden aus, sondern als 6stelligen Zählerstand. Die interne Uhr (Adressen 18,19 und 20 im Speicherbereich) zählt alle 1/50 Sekunde um eins hoch.

ATARINSIDE

USR
Format : USR(Adresse,n)
Beispiel: 550 A=USR(989,0)

Mit der USR-Funktion werden Maschinenspracheunterroutinen aufgerufen. Diese werden in einen bestimmten Speicherbereich des Computers geschrieben.

Der Vorteil der Maschinensprache liegt in der sehr viel höheren Verarbeitungsgeschwindigkeit gegenüber der Programmiersprache BASIC.

Mit Adresse geben Sie die Startadresse der Maschinenspracheroutine ein, mit n einen optionalen Wert. Dieser kann verschiedene Funktionen innerhalb der Unterroutine haben, z.B. als Zeiger auf ein Datenfeld oder um einen Parameter zu übertragen.

Vor dem USR muß immer eine Variable stehen. (Siehe auch Beispiel).
Beispielprogramm (Farbwechsel):

```
10 REM BEISPIEL ZUM TESTEN DER
20 REM USR FUNKTION
30 REM DIE MASCHINEROUTINE LAUTET
40 REM LDA #35 : STA 710 : RTS
100 A=0:T=0:COL=0:C=0
110 OPTION RESERVE 10
120 ADDR = VARPTR(RESERVE) :STARTADRESSE
130 FOR I=0 TO 5
140 READ A
150 POKE ADDR+I,A
160 NEXT
170 DATA &A9,&23,&8D,&C6,&02,&60
180 A=USR(ADDR,VARPTR(I)):STOP
```



# BESONDERHEITEN DES ATARI COMPUTER SYSTEMS 6

## GRAFIK ÜBERSICHT

Mit dem Grafik-Befehl wird jeweils einer der 12 möglichen Grafikmodi des ATARI Computer Systems angewählt. Dabei werden die Modi von 0 bis 8 mit dem Spezialbaustein CTIA erzeugt. Die dem Befehl GRAPHICS folgenden Ziffern müssen ganzzahlig und positiv sein. Der Grafikmodus 0 ist der sogenannte Textmodus, d.h. in ihm findet die Textdarstellung statt. Er wird beim Einschalten des System oder nach einem System Reset automatisch angewählt.

Die Grafikmodi 1 bis 8 benutzen geteilte Bildschirme. Im unteren Teil befindet sich ein Textfenster, in das vier Zeilen Text geschrieben werden können.

Die Grafikmodi 0,1 und 2 stellen Text in verschiedenen Größen dar. Bei Grafik 0 werden die Zeichen in normaler Größe dargestellt, bei Grafik 1 in doppelter Breite und bei Grafik 2 in doppelter Breite und doppelter Höhe. Grafikzeichen (CTRL-Taste) können bei Grafik 1 und 2 erst nach Umschalten der Adresse des Zeichensatzes dargestellt werden (POKE 756,226).

Grafik 3 bis Grafik 11 erlauben das PLOTen ( zeichnen ) einzelner Bildpunkte auf dem Bildschirm. Durch die Wahl des Grafikmodus wird die Größe dieser Bildpunkte und die Anzahl der möglichen Farben bestimmt. Es ist maximal die Darstellung von vier Farben auf einem Bildschirm möglich. Vier weitere Farben können durch die gleichzeitige Verwendung von PLAYER / MISSILE-Objekten erzielt werden (siehe Abschnitt 7).

Grafik 9 bis 11 sind nur möglich, wenn Ihr Computer mit einem GTIA Chip (Spezialbaustein) ausgestattet ist. Bei den für Europa hergestellten ATARI Computern ist dieser Baustein vorhanden. In Grafik 9 steht Ihnen eine Farbe in 16 Helligkeitsabstufungen zur Verfügung. Bei Grafik 10 neun Farben in acht Helligkeiten und bei Grafik 11 schließlich 16 Farben in einer Helligkeit.

ATARINSIDE

GRAPHICS
Format : GRAPHICS arithmetischer Ausdruck
Beispiele: GRAPHICS 2
100 GRAPHICS 5+16
120 GRAPHICS 1+32+16
500 GRAPHICS 8
120 GRAPHICS 0
320 GRAPHICS 18

Mit dem GRAPHICS-Befehl wird einer der Grafikmodi des ATARI Computers ausgewählt. In der Tabelle 6.1 können Sie sehen, welche Möglichkeiten die jeweiligen Grafikmodi bieten.

Alle Zeichen können im Grafikmodus 0 mit der PRINT-Anweisung dargestellt werden. Bei den Grafikmodi 1 bis 8 ist im unteren Bereich des Bildschirmes ein Textfenster vorhanden. Mit PRINT wird hier Text ausgegeben.

Bei GRAPHICS 1 und GRAPHICS 2 erfolgt die Ausgabe in das Grafikfenster durch die PRINT#6 Anweisung:

```
100 GRAPHICS 1
110 PRINT#6;AT(3,3);"GRAFIK FENSTER"
120 PRINT "TEXT-FENSTER"
```

Durch Addition von 16 (bei GRAPHICS 1 bis GRAPHICS 11) wird das Textfenster vermieden. Es steht Ihnen der gesamte Bildschirm zur Grafikdarstellung zur Verfügung. Falls Sie nun das obige Beispielprogramm starten, wird bei Zeile 120 automatisch in den Grafikmodus 0 zurückgeschaltet. Dies geschieht auch nach Beendigung eines Programms, das ausschließlich in einem anderen Grafikmodus lief. Dies kann z.B. durch die im folgenden Programm angefügte Zeile 140 verhindert werden:

```
110 GRAPHICS 2+16
120 PRINT#6,AT(3,3);"DIES IST GRAFIK"
130 PRINT#6,AT(10,5);"Z"
140 GOTO 140
```

Beenden des Programms ist nur durch BREAK möglich.

Normalerweise wird der Bildschirm durch Aufrufen eines Grafikmodus gelöscht. Dies läßt sich verhindern, indem 32 addiert wird.

Die Grafikmodi 3 bis 11 sind zum Zeichnen (PLOT) einzelner Bildpunkte geeignet. Sie bestimmen dabei, welche Farbe und Helligkeit die einzelnen Punkte haben. Ein Beipielprogramm befindet sich am Ende dieses Abschnitts.

Um wieder in den Textmodus zu gelangen, können Sie im Direktmodus GRAPHICS 0 eingeben.



Tabelle 6.1 GRAFIK MODI UND BILDSCHIRMFORMATE

| GRAFIK MODE | MODE TYP | SPALTEN | ZEILEN MIT TEXTFENSTER | ZEILEN OHNE TEXTFENSTER | ANZAHL FARBEN | SPEICHERBEDARF IN BYTE |
|---|---|---|---|---|---|---|
| CTIA | | | | | | |
| 0 | TEXT | 40 | - | 24 | 1-1/2 | 992 |
| 1 | TEXT | 20 | 20 | 24 | 5 | 674 |
| 2 | TEXT | 20 | 10 | 12 | 5 | 424 |
| 3 | GRAFIK | 40 | 20 | 24 | 4 | 434 |
| 4 | GRAFIK | 80 | 40 | 48 | 2 | 694 |
| 5 | GRAFIK | 80 | 40 | 48 | 4 | 1174 |
| 6 | GRAFIK | 160 | 80 | 96 | 2 | 2174 |
| 7 | GRAFIK | 160 | 80 | 96 | 4 | 4198 |
| 8 | GRAFIK | 320 | 160 | 192 | 1-1/2 | 8112 |
| CTIA | | | | | | |
| 9 | GRAFIK | 80 | - | 192 | 1 | 8112 |
| 10 | GRAFIK | 80 | - | 192 | 9 | 8112 |
| 11 | GRAFIK | 80 | - | 192 | 16 | 8112 |

GRAPHICS 3 bis 11 erlauben die Darstellung von einzelnen Bildpunkten. Die Größe dieser Punkte wird durch die Wahl des Modi bestimmt. Im Grafikmodus 3 sind die Grafikpunkte am größten, bei Grafik 8 am kleinsten.
Beispielprogramm:

```
10 INPUT "WELCHER GRAFIMODUS (3-8)";G
20 GRAPHICS G+16
30 COLOR 1
40 PLOT 5,5
50 FOR ZEIT = 1 TO 1900:NEXT
50 GOTO 10
```

COLOR (Farbe)
Format : COLOR n
Beispiel: 100 COLOR 4

Mit dem COLOR-Befehl legen Sie die Farbe des zu zeichnenden Punktes auf einen bestimmten Wert fest. Die Eingabe erfolgt durch die Angabe einer Zahl (n). Diese gibt das jeweilige Farbregister an.

Wie sich dieses Farbregister aufbaut können Sie aus der nachfolgenden Tabelle 6.2 ersehen. So entspricht zum Beispiel im Grafikmodus 3 das Farbregister 4 der Farbe orange, 5 der Farbe hellgrün und 6 der Farbe dunkelblau.

Sie können den COLOR-Befehl auch in Verbindung mit einem SETCOLOR-Befehl geben. Fügen Sie in obiges Beispielprogramm die Zeile 15 SETCOLOR 4,4,8 ein und starten Sie es. Sie werden sehen, daß das Farbregister vom SETCOLOR-Befehl beeinflußt wird.

ANMERKUNG: Bei der Arbeit mit GRAPHICS muß immer ein COLOR-Befehl gegeben werd, bevor das erste Mal geplottet wird.

ATARINSIDE

Tabelle 6.2 Farben, Grafikmodi, SETCOLOR und COLOR

| Farbe | MODE | Farb - register | COLOR n | Kommentar |
|---|---|---|---|---|
| | GRAPH.0 | 4 | | - |
| hellblau | | 5 | | Zeichenhelligkeit |
| dunkelblau | | 6 | | (wie Hintergrund) |
| | | 7 | | Zeichen |
| schwarz | Textmodus | 8 | =Rand | - |
| orange | | 4 | | Zeichen |
| hellgrün | GRAPH.1,2 | 5 | | Zeichen |
| dunkelblau | | 6 | | Zeichen |
| rot | | 7 | | Zeichen |
| schwarz | Textmodi | 8 | | Hintergrund, Rand |
| orange | | 4 | 1 | Bildpunkt |
| hellgrün | GRAPH.3,5,7 | 5 | 2 | Bildpunkt |
| dunkelblau | | 6 | 3 | Bildpunkt |
| | | 7 | - | - |
| schwarz | 4 Farbenmodi | 8 | 0 | Hintergrund, Rand |
| orange | GRAPH.4,6 | 4 | 1 | Bildpunkt |
| | | 5 | - | - |
| | | 6 | - | - |
| | | 7 | - | - |
| schwarz | 2 Farbenmodi | 8 | 0 | Hintergrund, Rand |
| | GRAPH.8 | 4 | - | - |
| hellblau | | 5 | 1 | - |
| dunkelblau | | 6 | 0 | - |
| | 1 Farbe / | 7 | - | - |
| schwarz | 2 Helligk. | 8 | - | Rand |
| schwarz | GRAPH.9 | 8 | 0 - 15 | Bildpunkte verschiedener Helligkeit |
| schwarz | GRAPH.10 | 0 | 0 | Bildpunkt |
| schwarz | | 1 | 1 | Bildpunkt |
| schwarz | | 2 | 2 | Bildpunkt |
| schwarz | | 3 | 3 | Bildpunkt |
| orange | | 4 | 4 | Bildpunkt |
| hellgrün | | 5 | 5 | Bildpunkt |
| dunkelblau | | 6 | 6 | Bildpunkt |
| rot | | 7 | 7 | Bildpunkt |
| schwarz | | 8 | 8 | Hintergrund |
| grau | GRAPH.11 | 8 | 0 - 15 | Bildpunkte |

ANMERKUNG: Die Farben der PLAYER / MISSILE-Objekte werden durch den SETCOLOR-Befehl bestimmt. Die Farbe des Registers entspricht der Farbe des jeweiligen PLAYER / MISSILE-Objektes, also Register 1 für PM 1, für PM 2 Register 2 usw.



SETCOLOR (setze Farbe)
Format : SETCOLOR Register,Farbe,Helligkeit
Beispiel: 100 SETCOLOR 5,4,10

Die SETCOLOR-Anweisung bestimmt die Farbe und die Helligkeit festgelegter Bildschirminhalte.

Die COLOR-Register 0,1,2 und 3 entsprechen den Farben der PLAYER / MISSILE-Objekten 0,1,2 und 3. Die Register 4,5,6 und 7 bestimmen die Farben bestimmter Bildschirmteile und das Register 8 bestimmt immer die Hintergrundfarbe.

Die Angabe der Farbe kann mit einer Zahl zwischen 0 und 15 erfolgen. (Siehe Tabelle 6.3). Der Helligkeitswert reicht von 0 bis 14. Dabei ist 0 der dunkelste und 14 der hellste Wert.

Tabelle 6.3 Die ATARI Farben

| Farbe | Nummer (dezimal) | Nummer (hexadez.) |
|---|---|---|
| grau | 0 | 0 |
| orange (gold) | 1 | 1 |
| orange | 2 | 2 |
| rot-orange | 3 | 3 |
| rosa | 4 | 4 |
| purpur | 5 | 5 |
| purpur-blau | 6 | 6 |
| azurblau | 7 | 7 |
| himmelblau | 8 | 8 |
| hellblau | 9 | 9 |
| türkis | 10 | A |
| grün-blau | 11 | B |
| grün | 12 | C |
| gelb-grün | 13 | D |
| orange-grün | 14 | E |
| hell orange | 15 | F |

PLOT / PLOT...TO (zeichne / zeichne bis zu)
Formate : PLOT X,Y
PLOT X,Y TO X1,Y1
Beispiele: 100 PLOT 12,9
120 PLOT 12,9 TO 3,3

Mit dieser Anweisung können Sie an irgendeine Stelle des Bildschirms einen Bildpunkt zeichnen (plotten). Dazu wird eine X,Y Koordinate angegeben. Die linke obere Ecke des Bildschirms ist immer die Koordinate 0,0. Dann verläuft die X Koordinate waagerecht und die Y Koordinate senkrecht über den Bildschirm.

In der Tabelle 6.1 können Sie ersehen, wieviele Bildpunkte die X Koordinate (Zeichen pro Zeile) und wieviele die Y Koordinate (Zeile) haben kann. Bei der Anwendung wird nach PLOT zunächst der X- und dann der Y-Wert angegeben. Mit der PLOT..TO-Anweisung wird eine Reihe von Punkten gezeichnet, sodaß sich eine Linie ergibt. Beispiel:
10 GRAPHICS 3+16: COLOR 1: PLOT 0,5 TO 15,5

ATARINSIDE

Hier ist ein Beispielprogramm, in dem die angesprochenen Anweisungen PLOT, COLOR und SETCOLOR verwendet wurden:

```
100 GRAPHICS 3+16
110 SETCOLOR 5,4,8
120 SETCOLOR 6,0,4
130 SETCOLOR 8,8,6
140 COLOR 1
150 PLOT 5,5 TO 10,5 TO 10,10 TO 5,10 TO 5,5
160 COLOR 2
170 PLOT 7,7 TO 12,12 TO 2,12 TO 7,7
180 COLOR 3
190 PLOT 2,7 TO 12,7
200 GOTO 200
```

FILL (füllen)
Format : FILL X,Y TO X1,Y1
Beispiel: 500 FILL 10,10 TO 5,5

Mit FILL wird ein bestimmter Bereich des Bildschirms mit der gewählten Farbe ausgefüllt. Dieses Ausfüllen wird unterbrochen, wenn eine Linie oder ein Bildpunkt angetroffen wird, der eine andere Farbe hat wie der Hintergrund.

Das Beispiel zeigt Ihnen, wie die FILL Anweisung arbeitet. Es werden drei Seiten eines Rechtecks erstellt. Die fehlende Seite wird von der FILL Anweisung gezeichnet. Dann wird dieses Rechteck mit einer bestimmten Farbe gefüllt.
Beispielprogramm:

```
10 GRAPHICS 5
20 SETCOLOR 4,12,8 !Register 4, grün, mittlere Helligkeit
30 COLOR 1 !Color 1 ist gleich mit SETCOLOR 4 bei GRAFIK 5
40 PLOT 5,5 TO 20,5 TO 20,20 TO 5,20
50 FILL 5,5 TO 5,20
60 END
```

Das Füllen des Rechtecks geschieht von der linken Seite bis zum Erreichen der rechten Linie .

CLS (Clear Screen / lösche Bildschirmwerte)
Format : CLS(Hintergrund Register Option)
Beispiele: CLS
100 CLS
110 GRAPHICS 3:CLS &C5
330 CLS 25

Mit der CLS-Anweisung wird bei Grafik 0 und Grafik 8 der Bildschirm gelöscht und die Helligkeit und Farbe des Randes ändert sich in den angegebenen Wert. Bei den anderen Grafikmodi wird die Hintergrundfarbe- und Helligkeit geändert. Geben Sie bei Grafik 0 einmal CLS 180 ein.



Tabelle 6.4 Zeichen in den Grafikmodi 1 und 2

| POKE 756,224 | POKE 756,226 | SETCOLOR 4 | SETCOLOR 5 | SETCOLOR 6 | SETCOLOR 7 |
|---|---|---|---|---|---|
| | | 32 | 0 | 160 | 128 |
| ! | | 33 | 1 | 161 | 129 |
| " | | 34 | 2 | 162 | 130 |
| # | | 35 | 3 | 163 | 131 |
| $ | | 36 | 4 | 164 | 132 |
| % | | 37 | 5 | 165 | 133 |
| & | | 38 | 6 | 166 | 134 |
| ' | | 39 | 7 | 167 | 135 |
| ( | | 40 | 8 | 168 | 136 |
| ) | | 41 | 9 | 169 | 137 |
| * | | 42 | 10 | 170 | 138 |
| + | | 43 | 11 | 171 | 139 |
| , | | 44 | 12 | 172 | 140 |
| - | | 45 | 13 | 173 | 141 |
| . | | 46 | 14 | 174 | 142 |
| / | | 47 | 15 | 175 | 143 |
| 0 | | 48 | 16 | 176 | 144 |
| 1 | | 49 | 17 | 177 | 145 |
| 2 | | 50 | 18 | 178 | 146 |
| 3 | | 51 | 19 | 179 | 147 |
| 4 | | 52 | 20 | 180 | 148 |
| 5 | | 53 | 21 | 181 | 149 |
| 6 | | 54 | 22 | 182 | 150 |
| 7 | | 55 | 23 | 183 | 151 |
| 8 | | 56 | 24 | 184 | 152 |
| 9 | | 57 | 25 | 185 | 153 |
| : | | 58 | 26 | 186 | 154 |
| ; | | 59 | 27 | 187 | 155 |

ATARINSIDE

Tabelle 6.4 Zeichen in den Grafikmodi 1 und 2

| POKE 756,224 | POKE 756,226 | SETCOLOR 4 | SETCOLOR 5 | SETCOLOR 6 | SETCOLOR 7 |
|---|---|---|---|---|---|
| < | ↑ | 60 | 28 | 188 | 156 |
| = | ↓ | 61 | 29 | 189 | 167 |
| > | ← | 62 | 30 | 190 | 168 |
| ? | → | 63 | 31 | 191 | 169 |
| @ | ` | 64 | 96 | 192 | 224 |
| A | a | 65 | 97 | 193 | 225 |
| B | b | 66 | 98 | 194 | 226 |
| C | c | 67 | 99 | 195 | 227 |
| D | d | 68 | 100 | 196 | 228 |
| E | e | 69 | 101 | 197 | 229 |
| F | f | 70 | 102 | 198 | 230 |
| G | g | 71 | 103 | 199 | 231 |
| H | h | 72 | 104 | 200 | 232 |
| I | i | 73 | 105 | 201 | 233 |
| J | j | 74 | 106 | 202 | 234 |
| K | k | 75 | 107 | 203 | 235 |
| L | l | 76 | 108 | 204 | 236 |
| M | m | 77 | 109 | 205 | 237 |
| N | n | 78 | 110 | 206 | 238 |
| O | o | 79 | 111 | 207 | 239 |
| P | p | 80 | 112 | 208 | 240 |
| Q | q | 81 | 113 | 209 | 241 |
| R | r | 82 | 114 | 210 | 242 |
| S | s | 83 | 115 | 211 | 243 |
| T | t | 84 | 116 | 212 | 244 |
| U | u | 85 | 117 | 213 | 245 |
| V | v | 86 | 118 | 214 | 246 |
| W | w | 87 | 119 | 215 | 247 |



Tabelle 6.4 Zeichen in den Grafikmodi 1 und 2

| POKE 756,224 | POKE 756,226 | SETCOLOR 4 | SETCOLOR 5 | SETCOLOR 6 | SETCOLOR 7 |
|---|---|---|---|---|---|
| X | x | 88 | 120 | 216 | 248 |
| Y | y | 89 | 121 | 217 | 249 |
| Z | z | 90 | 122 | 218 | 250 |
| [ | ♣ | 91 | 123 | 219 | 251 |
| \ | \| | 92 | 124 | 220 | 252 |
| ] | ↖ | 93 | 125 | 221 | 253 |
| ^ | ◀ | 94 | 126 | 222 | 254 |
| _ | ▶ | 95 | 127 | 223 | 255 |

Beispielprogramm:

In diesem Programm wird die Ausgabe der Zeichen in den Grafokmodi 1 und 2 demonstriert. Es wird der normale Zeichensatz und der Spezialzeichensatz (POKE 756,226) angesprochen. Zum Abbrechen des Programms drücken Sie bitte die BREAK-Taste.

```
2 REM SCHREIBMASCHINE
10 GRAPHICS 2
20 SETCOLOR 4,0,0
30 PRINT "GEBEN SIE GRUEN/BLAU/ROT (G/B/R) EIN"
40 INPUT "UND DRUECKEN SIE RETURN ";C$
50 IF C$="G" THEN K=32
60 IF C$="B" THEN K=128
70 IF C$="R" THEN K=160
80 PRINT "GEBEN SIE GROSS/KLEIN (G/K) EIN"
90 INPUT "UND DRUECKEN SIE RETURN ";B$
100 IF B$="G" THEN 120
110 POKE 756,226
120 PRINT "NUN DRUECKEN SIE BUCHSTABEN UND"
125 PRINT " CTRL-TASTEN."
130 A$=INKEY$
140 IF A$="" THEN 130
150 A=ASC(A$)+K:32=GRUEN, 128 IST BLAU, 160 IST ROT
160 PRINT A
170 PRINT #6,CHR$(A);
180 GOTO 130
```

```
100 REM BLINKEN
110 GRAPHICS 16+2
120 X=RND(36)
130 ON ERROR GOTO 150
140 PRINT#6,TAB(X);"*"
146 GOTO 120
150 GRAPHICS 32+16+2
160 RESUME
```

ATARINSIDE

Das folgende kurze Programm demonstriert die in Tabelle 6.4 angegebenen Zeichen. Es gibt im Textfenster die Code-Nummer des Zeichens aus und das Zeichen selbst im Grafikfenster. Immer wenn Sie die RETURN-Taste drücken, wird ein neues Zeichen ausgegeben. Durch die Anweisungen SETCOLOR 8,0,0 und SETCOLOR 4,0,0 wird verhindert, daß bei Kleinbuchstaben der Bildschirm voll Herzen gedruckt wird. Ein anderer Weg wäre das Umschreiben des Zeichensatzes. Im Programm wird der Spezialzeichensatz ausgegeben. Wenn Sie den normalen Zeichensatz sehen möchten, löschen Sie bitte die Zeile 20.

```
10 GRAPHICS 2
20 POKE 756,226
30 SETCOLOR 8,0,0
40 SETCOLOR 4,0,0
50 SETCOLOR 5,4,6
60 SETCOLOR 6,12,2
70 SETCOLOR 7,9,6
80 A$=INKEY$
90 IF A$="" THEN 80
100 ON ERROR GOTO 150
110 PRINT #6,AT(6,6);CHR$(X)
120 PRINT X
130 X=X+1
140 GOTO 80
150 RUN ! WIEDERHOLUNG WENN X=256
```

DER SOUND BEFEHL

Format : SOUND Stimme, Frequenz, Verzerrung, Lautstärke, Dauer

Beispiele: 120 SOUND 2,204,10,12,244
125 SOUND 0,122,8,10

Die Stimme gibt an, welchen der vier Tongeneratoren Sie benutzen möchten. Es ist natürlich auch möglich, alle vier gleichzeitig zu benutzen. Dabei ist 0 der erste Tongenerator, 1 der zweite usw.

Die Frequenz kann zwischen 0 und 255 liegen (Tabelle 6.5). Die Verzerrung kann als gerade Zahl von 0 bis 14 angegeben werden. Dabei entspricht 10 einem reinen Ton.

Die Lautstärke wird mit einer Zahl von 0 bis 15 angegeben. 0 ist dabei der leiseste und 15 der lauteste Ton. 8 entspricht der normalen Lautstärke. Die Summe der Lautstärken aller im Betrieb genommenen Tongeneratoren sollte 24 nicht überschreiten, da sonst Verzerrungen auftreten (bei den Modellen 400/800 darf der Wert auch 32 erreichen).

Die Dauer wird in 1/50 Sekunden angegeben. Wenn die Dauer nicht festgelegt wird, erklingt der Ton, bis er durch einen END- oder RUN-Befehl unterbrochen wird. Eine andere Möglichkeit, den Ton abzubrechen, ist die Eingabe des betreffenden Tongenerators gefolgt von Nullen (SOUND n,0,0,0) oder das Drücken der BREAK-Taste.



Tabelle 6.5 Frequenzwerte und Noten

| Tonart | Noten | Hexzahl | Dezimalzahl |
|---|---|---|---|
| Hohe Töne | C | 1D | 29 |
| | B | 1F | 31 |
| | A#/Bb | 21 | 33 |
| | A | 23 | 35 |
| | G#/Ab | 25 | 37 |
| | G | 28 | 40 |
| | F#/Gb | 2A | 42 |
| | F | 2D | 45 |
| | E | 2F | 47 |
| | D#/E | 32 | 50 |
| | D | 35 | 53 |
| | C#/Db | 39 | 57 |
| | C | 3C | 60 |
| | B | 40 | 64 |
| | A#/B | 44 | 68 |
| | A | 48 | 72 |
| | G#/Ab | 4C | 76 |
| | G | 51 | 81 |
| | F#/Gb | 55 | 85 |
| | F | 5B | 91 |
| | E | 60 | 96 |
| | D#/Eb | 66 | 102 |
| | D | 6C | 108 |
| | C#/Db | 72 | 114 |
| mittlere Töne | C | 79 | 121 |
| | B | 80 | 128 |
| | A#/Eb | 88 | 136 |
| | A | 90 | 144 |
| | G#/Ab | 99 | 153 |
| | G | A2 | 162 |
| | F#/Gb | AD | 173 |
| | F | B6 | 182 |
| tiefe Töne | E | C1 | 193 |
| | D#/Eb | CC | 204 |
| | D | D9 | 217 |
| | C#/Db | E6 | 230 |
| | C | F3 | 243 |

ATARINSIDE

Beispielprogramm:

```
10 GRAPHICS 2+16
20 SETCOLOR 4,8,4
30 PRINT#6,AT(3,3)
40 FOR ZEIT = 1 TO 1000: NEXT
50 GRAPHICS 2+16
60 PRINT#6,AT(3,3);"START-BASIS"
70 FOR ZEIT = 1 TO 1000: NEXT
80 GRAPHICS 0
90 POKE 752,1
100 SETCOLOR 6,0,0
110 FOR T=1 TO 24: PRINT "": NEXT
120 PRINT TAB(11);CHR$(8);CHR$(10)
130 PRINT TAB(11);CHR$(22);CHR$(2)
140 PRINT TAB(11);CHR$(22);CHR$(2)
150 PRINT TAB(11);CHR$(13);CHR$(13)
160 PRINT TAB(11);CHR$(6);CHR$(7)
170 FOR VOL=15 TO 0 STEP-1
180 SOUND 2,77,8,VOL
190 PRINT CHR$(155):RAKETE HEBT AB
200 FOR R=1 TO 200 : NEXT R
210 NEXT VOL
220 END
```

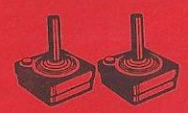
Dieses Programm demonstriert wie man durch Ändern der Lautstärke einen bestimmten Effekt erreichen kann ( akustisches Entfernen der Rakete).

## SPIEL-STEUERUNGEN

Im ATARI Microsoft BASIC II werden die Werte der Spielsteuerungen mit PEEK aus den entsprechenden Adressen gelesen. Die Regler, Joysticks etc. übertragen die Werte über die am Computer vorhandenen Buchsen.

Eine komplette Liste aller wichtigen Adressen im Speicher ist in Anlage E aufgeführt.

Sie können die Anweisung DEF verwenden, um eigene Steuerkommandos zu definieren (Siehe S.27 ff).



JOYSTICK

PADDLE



## PADDLES (Drehregler)

Das folgende Beispielprogramm liest und schreibt den jeweiligen Wert des Drehreglers 0 auf den Bildschirm (erster Drehregler in Buchse 1). Diese PEEK-Anweisung kann auch mit anderen Funktionen benutzt werden, z.B. zum Erzeugen von Tönen, Grafiken o.ä.

Der Drehregler erzeugt Werte zwischen 1 und 228. Durch Rechtsdrehung des Reglers wird der Wert erhöht, bis er 228 erreicht.

```
10 DREHREGLER(0)=624
20 PRINT PEEK(DREHREGLER(0))
30 GOTO 20
```

Jedem Drehregler ist eine Adresse im Speicher zugeordnet:

```
DREHREGLER(0)=624
DREHREGLER(1)=625
DREHREGLER(2)=626
DREHREGLER(3)=627
DREHREGLER(4)=628
DREHREGLER(5)=629
DREHREGLER(6)=630
DREHREGLER(7)=631
```

An jedem Drehregler befindet sich zusätzlich ein roter Knopf. Auch dieser Knopf wird gelesen. Ist er gedrückt, enthält die entsprechende Adresse eine 0, andernfalls eine 1.

```
10 DRKNOPF(0)=&27C
20 PRINT PEEK(DRKNOPF(0))
30 GOTO 20
```

Die Adressen der Druckknöpfe der jeweiligen Drehregler:

```
DRKNOPF(0)=636
DRKNOPF(1)=637
DRKNOPF(2)=638
DRKNOPF(3)=639
DRKNOPF(4)=640
DRKNOPF(5)=641
DRKNOPF(6)=642
DRKNOPF(7)=643
```

## JOYSTICKS (Steuerknüppel)

Das Lesen der Adressen der Steuerknüppel erfolgt auf die gleiche Weise wie das Lesen der Drehregler-Adressen. Die Steuerknüppel sind von links nach rechts durchgehend nummeriert (0-3).

Beispiel (Steuerknüppel 0):

```
10 STICK(0)=632
20 PRINT PEEK(STICK(0))
30 GOTO 20
```

ATARINSIDE

Die Adressen der Steuerknüppel:
STICK(0)=632
STICK(1)=633
STICK(2)=634
STICK(3)=635

Abb. 6.2 zeigt welcher Wert bei welcher Stellung des Steuerknüppels in der Adresse steht:

Die Feuerknöpfe der Steuerknüppel werden auf die gleiche Weise gelesen wie die der Drehregler:

```
10 FKNOPF(0)=644
20 PRINT PEEK(FKNOPF(0))
30 GOTO 20
```

Die Adressen der Feuerknöpfe:
FKNOPF(0)=644
FKNOPF(1)=645
FKNOPF(2)=646
FKNOPF(3)=647

```
10 IF PEEK(644)=0 THEN PRINT "FEUERKNOPF GEDRUECKT"
20 IF PEEK(644)=1 THEN CLS
30 GOTO 20
```

## TASTEN MIT SPEZIALFUNKTION

Das folgende Programm liest die Adressen der drei Zusatztasten (OPTION, SELECT und START).

```
10 POKE 53279,0
20 PRINT PEEK(53279)
30 GOTO 20
```

Die Adresse, die gelesen wird, ist 53279. Diese Adresse nimmt bei jedem Tastendruck einen speziellen Wert an. Wenn die Taste losgelassen wird, enthält die Adresse immer den Wert 7.
Die Werte und ihre Bedeutung:
7 = keine Taste gedrückt
6 = START-Taste gedrückt
5 = SELECT-Taste gedrückt
3 = OPTION-Taste gedrückt



# PLAYER MISSILE GRAFIK 7

Der ATARI Computer verfügt über eine ausgezeichnete Möglichkeit spezielle Figuren zu erstellen und diese zu bewegen. Diese Fähigkeit wird PLAYER / MISSILE-Grafik genannt.

Die Begriffe PLAYER und MISSILE sind von der beweglichen Grafik der ATARI Video Spiele übernommen worden. Für die Erstellung und Bewegung der PM-Objekte kann ein bestimmter Speicherbereich freigehalten werden. Dort werden die Figuren als Binärmuster abgelegt. Es können 4 Players und 4 Missiles erstellt werden. Die vier Missiles können auch zu einem fünften Player zusammengefaßt werden. Im Normalfall ist jedoch jedem Player ein Missile zugeordnet.

Ein Player kann in jedem Grafikmodus auf dem Bildschirm dargestellt werden. Der Bildschirminhalt wird nicht verändert.

In der Abb. 7.2 ist ein Player zu sehen. Überall wo auf der Abbildung eine 1 ist, erscheint ein Bildpunkt auf dem Bildschirm, wo eine 0 ist, wird auch auf dem Bildschirm nichts ausgegeben. Player können unter Benutzung von OPTION(PLM1) mit einfacher oder unter Benutzung von OPTION(PLM2) mit doppelter Auflösung erstellt werden. Wenn Sie die einfache Auflösung wählen, entspricht jedes Byte (8 bit = 8 Bildpunkte) einer Bildschirmzeile (256 Bytes pro Player). Bei doppelter Auflösung entsprechen jedem Byte 2 Bildschirmzeilen (128 Bytes pro Player). Die Auflösung bezieht sich auf alle im Programm verwendeten Player. Ein Mischen ist nicht möglich.

Die folgenden Anweisungen und Befehle helfen Ihnen bei der Erstellung von Playern und Missiles:
MOVE
OPTION(PLM1 bzw. PLM2)
VARPTR(PLM1 bzw. PLM2)
SETCOLOR 0,1,2 bzw. 3

## WIE HILFT ATARI MICROSOFT BASIC II PLAYER MISSILE GRAFIKEN ZU ERSTELLEN ?

Die MOVE-Anweisung wird benutzt um das Player Missile-Objekt auf und ab zu bewegen. Um dies zu verdeutlichen, stellen Sie sich vor, Sie hätten vor sich einen schmalen Papierstreifen liegen. Auf diesen malen Sie nun in seiner gesamten Breite ein Objekt. Um es nun zu bewegen, müßten Sie das Objekt ausradieren und ober- oder unterhalb neu aufzeichnen.

Der MOVE-Befehl arbeitet ähnlich, wenn auch wesentlich schneller.

Die Rechts-/ Linksbewegung eines PM-Objektes ist leichter zu programmieren. Um bei dem Papierstreifen zu bleiben: Sie brauchen ihn nur nach rechts und links zu bewegen.

ATARINSIDE

Hinter die MOVE-Anweisung schreiben Sie zunächst die Adresse, an der das PM-Objekt sich befindet (die Adresse erfahren Sie unter Verwendung der Anweisung VARPTR(PLM1 oder PLM2)). Dann geben Sie die Adresse an, an die sich das PM-Objekt bewegen soll. Zuletzt geben Sie noch die Anzahl der Byte an, die bewegt werden sollen (MOVE von Adresse, zur Adresse, Anzahl Bytes).

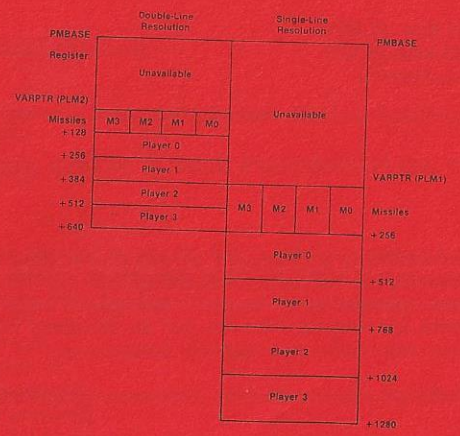
Durch die Anweisung OPTION(PLM1) löschen und reservieren Sie einen Speicherbereich zur Erstellung eines PM-Objektes einfacher Auflösung. OPTION(PLM2) verwendet man für PM-Objekte mit doppelter Auflösung.

VARPTR(PLM1 oder PLM2) gibt Ihnen an, wo dieser reservierte Speicherbereich beginnt. Diese Angabe benötigen Sie, um Players zu erstellen und dann zu bewegen. Ein Beispiel:

Die Startadresse (obere Ecke des Bildschirms) des Player(0) ist VARPTR(PLM2)+128 (siehe hierzu auch Abb 7.1: Player(0) beginnt bei VARPTR(PLM2)+128 bei doppelter Auflösung). Wenn Sie nun einen durchgehenden Strich in die Mitte des Player zeichnen wollen geben Sie ein: POKE VARPTR(PLM2)+192,255.

Die SETCOLOR-Anweisung gibt die Farbe des Players an. Das Missile hat immer die gleiche Farbe wie sein Player. Sie erinnern sich ? Die Farbregister 0,1,2 und 3 bestimmen die Farben der PM-Objekte 0,1,2 und 3.

In die Adresse 559 muß noch der Wert 62 (einfache Auflösung) oder 46 (doppelte Auflösung) gePOKEt werden. Zum "Anschalten" der PM Grafik geben Sie ein POKE 53277,3.






## ERSTELLEN EINES PLAYERS AUF PAPIER

Nehmen Sie ein Blatt kariertes Papier und zeichnen Sie die von Ihnen gewünschte Figur in einem Raster von maximal 8 mal 8 Kästchen (die Höhe muß selbstverständlich nicht 8 betragen, doch haben wir unser Beispiel auf 8x8 Kästchen ausgelegt). Dann werten Sie diese Figur aus, wie das in der folgenden Abbildung getan wurde.

| Graphic Representation | Binary Representation | Hexadecimal Representation | Decimal Representation |
|---|---|---|---|
| | 00011000 | 18 | 24 |
| | 00011000 | 18 | 24 |
| | 00100100 | 24 | 36 |
| | 00100100 | 24 | 36 |
| | 01000010 | 42 | 66 |
| | 01000010 | 42 | 66 |
| | 10000001 | 81 | 129 |
| | 10000001 | 81 | 129 |

Abbildung 7-2 AUSWERTUNG EINES PLAYERS

In der Abbildung sehen Sie ein auf dem Kopf stehendes V. Diese Abbildung entspricht einem Player. Nun geben Sie die ermittelten Werte ein. (POKE VARPTR(PLM1)+128,24 bzw. POKE VARPTR(PLM1)+256,24 usw. ).

## FARBENBESTIMMUNG

Der ATARI-Computer besitzt neun Register zur Bestimmung der Farbwerte der PM-Objekte, des Spielfeldes und des Hintergrunds.

Tabelle 7.1 SETCOLOR-Register-Belegung

| SETCOLOR | Funktion |
|---|---|
| SETCOLOR 0,Farbe,Helligkeit | Farbe / Helligkeit von PM(0) |
| SETCOLOR 1,Farbe,Helligkeit | Farbe / Helligkeit von PM(1) |
| SETCOLOR 2,Farbe,Helligkeit | Farbe / Helligkeit von PM(2) |
| SETCOLOR 3,Farbe,Helligkeit | Farbe / Helligkeit von PM(3) |
| SETCOLOR 4,Farbe,Helligkeit | Farbe / Helligkeit von Spielfeld(0) |
| SETCOLOR 5,Farbe,Helligkeit | Farbe / Helligkeit von Spielfeld(1) |
| SETCOLOR 6,Farbe,Helligkeit | Farbe / Helligkeit von Spielfeld(2) |
| SETCOLOR 7,Farbe,Helligkeit | Farbe / Helligkeit von Spielfeld(3) |
| SETCOLOR 8,Farbe,Helligkeit | Farbe / Helligkeit vom Hintergrund |

Die Player sind unabhängig vom Spielfeld und von anderen Playern. Wenn Sie aus den vier Missiles einen fünften Player machen, erhält er seine Farbe durch das Farbregister 3.

Wenn Sie den SETCOLOR-Befehl verwenden, geben Sie bitte das Register, die Farbe und deren Helligkeit an. Sehen Sie sich dazu auch die Zeilen 20 und 110 im Beispielprogramm auf Seite 74 an.

Die Farbregister arbeiten auch unabhängig voneinander. Allerdings können nicht alle Register gleichzeitig in allen Grafikmodi benutzt werden. Vergleichen Sie dazu den Abschnitt 6, GRAFIK.

ATARIINSIDE

## GRößENFESTLEGUNG

Beim ATARI-Computer sind fünf Register zum Bestimmen der Größe vorhanden; vier für die Player und eins für sämtliche Missiles.

Die Register und ihre Adressen im Speicherbereich:

Tabelle 7.2 Register zur Größenbestimmung

| Register Name | Adresse hexadezimal | Adresse dezimal | Funktion |
|---|---|---|---|
| SIZEP0 | D008 | 53256 | Größe von Player(0) |
| SIZEP1 | D009 | 53257 | Größe von Player(1) |
| SIZEP2 | D00A | 53258 | Größe von Player(2) |
| SIZEP3 | D00B | 53259 | Größe von Player(3) |
| SIZEM | D00C | 53260 | Größe aller Missiles |

Die Größenfestlegung erlaubt Ihnen, Player oder Missiles auch in doppelter oder vierfacher Breite darzustellen. POKEn Sie in die entsprechende Adresse den Wert:

0 oder 2 für normale Breite
1 für doppelte Breite
3 für vierfache Breite

Beachten Sie Zeile 80 im Beispielprogramm auf Seite 74.

## AUSGANGSPOSITION UND BEWEGUNG

Vertikal:

Die vertikale Position wird festgelegt, wenn Sie die Adresse des PM-Objektes im Speicher bestimmen. Je niedriger das PM-Objekt im RAM-Bereich plaziert ist, umso höher erfolgt die Darstellung auf dem Bildschirm (siehe auch Zeile 120 und 200 im Beispielprogramm).

Zum Bewegen von Player-Missile-Objekten benutzen Sie die MOVE-Anweisung. Da der MOVE-Befehl die alte Position des Players nicht selbständig löscht, muß das erste und letzte Byte des Players eine 0 enthalten. Dadurch löscht der Player das auf der alten Position stehende Byte bei der Auf- oder Abwärtsbewegung selbst. Stellen Sie sich das bitte so vor, als wenn Sie einen Bleistift hätten, an dem über und unter der Mine ein Radiergummi befestigt ist. Wenn Sie nun auf einem Blatt Papier senkrechte Striche ziehen, sehen Sie immer nur den von der Mine erzeugten Punkt, unseren Player.

Alle Bytes des Players müssen bewegt werden. Danach wird ein evtl. vorhandener Zähler auf- oder abgezählt (z.B. die Variable Y. Siehe auch dazu Zeile 360 im Beispielprogramm).

Horizontal

Die horizontale Position wird durch das POKEn in die entsprechende Adresse des Horizontal-Positions-Register erreicht (Tabelle 7.3).



Tabelle 7.3 Player-Missile-Horizontal-Positions-Register

| Register Name | Adresse hexadezimal | Adresse dezimal | Funktion |
|---|---|---|---|
| HPOSP0 | D000 | 53248 | Position Player(0) |
| HPOSP1 | D001 | 53249 | Position Player(1) |
| HPOSP2 | D002 | 53250 | Position Player(2) |
| HPOSP3 | D003 | 53251 | Position Player(3) |
| HPOSM0 | D004 | 53252 | Position Missile(0) |
| HPOSM1 | D005 | 53253 | Position Missile(1) |
| HPOSM2 | D006 | 53254 | Positiom Missile(2) |
| HPOSM3 | D007 | 53255 | Position Missile(3) |

Zum Setzen der horizontalen Position eines Players oder Missiles POKEn Sie einfach den gewünschten Wert in die betreffende Adresse. Der Wert kann zwischen 0 und 255 liegen. Allerdings ist das P-Objekt in der Lage, nach rechts und links aus dem Bildschirm zu wandern. Der sichtbare Bereich liegt bei einem Wert zwischen 60 und 200.

Diagonal

Eine diagonale Bewegung ist durch die Kombination von vertikaler und horizontaler Bewegung möglich. Geben Sie dabei erst die neue horizontale Position an und dann die vertikale Position.

## PRIORITATEN FESTLEGEN

Das Prioritäts-Kontroll-Register liegt bei der Adresse 53275 (hex D01B mit OS-Schattenregister &26F) im Speicher. Damit können Sie bestimmen, welche PM-Objekte und welche Spielfeldteile hintereinander verschwinden. Außerdem wird es benötigt, um einen fünften Player aus den 4 Missiles zu bilden.

Priorität auswählen

Dies geschieht wieder durch POKEn. Allerdings wird hier in die "Schattenadresse" des Priorität-Kontroll-Registers gePOKEt. Diese liegt bei 623, (hex 26F).

POKE 623,1 = alle Player vor der normalen Grafik (Spielfeld)
POKE 623,2 = Player 0 und Player 1 vor dem Spielfeld, alle Spielfelder vor Player 2 und Player 3 .
POKE 623,4 = alle Spielfelder vor den Playern
POKE 623,8 = Spielfelder 0 und 1 vor allen Player, alle Player vor Spielfeldern 2 und 3

Bilden des fünften Players

Sie können aus den vier Missiles einen funften Player bilden. Die Farbe wird durch das Farbregister 3 (Adresse 711, hex 2C7) bestimmt. Um den fünften Player horizontal zu bewegen, muß in alle Horizontal-Positions-Register der vier Missiles der gleiche Wert gePOKEt werden (53252 bis 53255, hex D004 bis D007).

ATARI INSIDE

## BERÜHRUNG FESTSTELLEN

Durch Abfrage der Kollisions-Kontroll-Register können Sie feststellen, ob zwischen den Player / Missiles und anderen Grafikobjekten ein Zusammenstoß erfolgte.

Tabelle 7.4 Kollisons-Kontroll-Register

| Register Name | Adresse hexadezimal | Adresse dezimal | Funktion |
|---|---|---|---|
| M0PF | D000 | 53248 | Missile(0)/Spielfeld |
| M1PF | D001 | 53249 | Missile(1)/Spielfeld |
| M2PF | D002 | 53250 | Missile(2)/Spielfeld |
| M3PF | D003 | 53251 | Missile(3)/Spielfeld |
| P0PF | D004 | 53252 | Player(0)/Spielfeld |
| P1PF | D005 | 53253 | Player(1)/Spielfeld |
| P2PF | D006 | 53254 | Player(2)/Spielfeld |
| P3PF | D007 | 53255 | Player(3)/Spielfeld |
| M0PL | D008 | 53256 | Missile(0)/Player |
| M1PL | D009 | 53257 | Missile(1)/Player |
| M2PL | D00A | 53258 | Missile(2)/Player |
| M3PL | D00B | 53259 | Missile(3)/Player |
| P0PL | D00C | 53260 | Player(0)/Player |
| P1PL | D00D | 53261 | Player(1)/Player |
| P2PL | D00E | 53262 | Player(2)/Player |
| P3PL | D00F | 53263 | Player(3)/Player |

Sie werden bemerken, daß ein Teil dieser Adressen bereits in der Tabelle 7.3 angeführt wurde. Aber um eine Kollision festzustellen, werden nur die ersten vier Bit dieser Register benötigt. Nimmt eins dieser vier Bit den Wert logisch 1 an, ist ein Zusammenstoß erfolgt. Diese 1 wird gelesen und bleibt bestehen, bis Sie durch einen POKE-Befehl wieder gelöscht wird (POKE 53278,0 bzw. POKE &D01E,0).

## PLAYER-MISSILE-BEISPIELPROGRAMM

Das folgende Beispielprogramm erstellt einen Player (Raumschiff) und läßt ihn Missiles verschießen. Gelenkt wird er durch einen Steuerknüppel, der in die Buchse 1 des ATARI-Computers eingesteckt wird.

LISTING

```
05 REM PLAYER MIT DOPPELTER AUFLOESUNG
10 GRAPHICS 8+16
20 SETCOLOR 6,0,0
30 X = 130
40 Y = 70
50 STICK0 = &278
60 OPTION PLM2
70 POKE 559,46
80 POKE &D00C,1
90 POKE &D01D,3
100 POKE &D000,X
110 SETCOLOR 0,3,10
120 FOR J = VARPTR(PLM2)+128+Y TO VARPTR(PLM2)+135+Y:READ
A:POKE J,A
```



```
125 NEXT J
130 DATA 0,129,153,189,255,189,153,0
140 IF PEEK(&D010) = 1 THEN 220
150 SOUND 0,ZZ0,12,15,INT(X/30)
160 ZAP=X
170 POKE VARPTR(PLM2)+4+Y,3
180 POKE &D004,ZAP
190 ZAP = ZAP-12
200 IF ZAP<12 THEN POKE VARPTR(PLM2)+4+Y,0:GOTO 220 ELSE 180
210 ! STEUERKNUEPPEL BEWEGUNG
220 A = PEEK(STICK0):IF A=15 THEN 140
230 IF A=11 THEN X=X-1
240 IF A=7 THEN X=X+1
250 POKE &D000,X
260 IF A=14 THEN 350:AUF
270 IF A=13 THEN 390:AB
280 ! VERTIKALE BEWEGUNG
290 IF A=10 THEN X=X-1:POKE &D000,X:GOTO 350
300 IF A=6 THEN X=X+1:POKE &D000,X:GOTO 350
310 IF A=9 THEN X=X-1:POKE &D000,X:GOTO 390
320 IF A=5 THEN X=X+1:POKE &D000,X:GOTO 390
330 GOTO 140
340 ! AUF
350 MOVE VARPTR(PLM2)+128+Y,VARPTR(PLM2)+128+(Y-1),8
360 Y=Y-1
370 GOTO 140
380 ! AB
390 MOVE VARPTR(PLM2)+128+(Y-1),VARPTR(LM2)+128+Y,8
400 Y=Y+1
410 GOTO 140
420 STOP
430 END
```

## KOMMENTARE ZUM PROGRAMM

ZEILE 10: es wird eine hochauflösende Grafik ohne Textfenster gewählt. Player Missile-Grafiken können in allen Grafikmodi erstellt und verwendet werden.

ZEILE 20: bestimmt die Farbe des Hintergrunds. 6 ist das Farbregister für den Hintergrund, 0 ist die Farbe schwarz mit der Helligkeit 0, also der dunkelste Wert.

ZEILEN 30,40: hier wird die X (horizontale) und Y (vertikale) Ausgangsposition des Players bestimmt.

ZEILE 50: weist der Variablen STICK0 den Wert &278 (dezimal 632) zu. (Adresse des Steuerknüppels(0)).

ZEILE 60: reserviert im Speicher Platz zur Erstellung eines Players mit doppelter Auflösung.

ATARINSIDE

ZEILE 70: bereitet das Register "direkter Speicherzugriff" für einen Player mit doppelter Auflösung vor.

ANMERKUNG: nach Ansprechen dieser Adresse werden die Player-Missile-Objekte aus dem reservierten Speicherbereich geladen.

ZEILE 80: hier wird festgelegt, daß der Player doppelte Breite haben soll (Player-Größen-Kontroll-Register).

ZEILE 90: schaltet die Player Missile-Grafik an. Mit POKE &D01D,0 wird ausgeschaltet (Player werden allerdings nicht gelöscht, das heißt sie bleiben auf dem Bildschirm sichtbar).

ZEILE 100: POKEt die horizontale Position des Player(0) in das Horizontal-Kontroll-Register.

ZEILE 120-125: setzt die Variable (VARPTR(PLM2)) an die Player Missile-Startadresse im Speicher. Die Daten aus der Zeile 130 werden in den reservierten Speichbereich gePOKEt.

ZEILE 140: teilt dem Computer mit, ob der Feuerknopf des Steuerknüppels gedrückt wurde. Enthält die Adresse hexD010 den Wert 1 ist der Knopf nicht gedrückt und die Programmausführung wird mit Zeile 220 fortgesetzt. Ist der Knopf gedrückt, steht in der Adresse eine 0; die Zeilen 150 bis 200 werden auch ausgeführt.

ZEILE 150: erzeugt ein Geräusch, wenn der Feuerknopf gedrückt ist. Der komplette SOUND-Befehl besteht aus:
1 dem angesprochenen Tongenerator (0 bis 3). Es können alle vier Tongeneratoren gleichzeitig angesprochen werden.
2 dem gewähltem Ton entsprechend der Tabelle 7.2 oder einem eigenen Geräusch.
3 der Angabe des Verzerrungsgrades mit einem Wert zwischen 0 und 14. Eine 10 gibt einen klaren, reinen Ton.
4 der Angabe eines Lautstärkewertes. Dieser kann zwischen 0 und 15 liegen (0 = leise, 15 = laut).
5 der Dauer des Tons in 1/50 Sekunden

ZEILE 160: bestimmt die horizontale Position des Missile (ZAP) entsprechend der Position des Players (X).

ZEILE 170: schaltet die Darstellung des Missile an (3 schaltet ein, 0 schaltet aus).

ZEILE 180: POKEt die horizontale Position des Missile in das Missile(0)-Horizontal-Register (&D004).

ZEILE 190: zählt die Variable ZAP um 12 herunter, um eine horizontale Schußlinie zu erhalten.



ZEILE 200: wenn die horizontale Position des Missiles kleiner als 12 ist, verschwindet er am linken Bildschirmrand. Um das zu vermeiden, werden in den reservierten Speicherbereich des Missiles Nullen gePOKEt. Danach wird die Zeile 220 angesprungen. Ist die Position 12 oder größer, wird die neue Position in Zeile 180 in die Adresse D004 gePOKEt und die Variable ZAP um 12 erniedrigt.

ZEILE 220: hier wird dem Computer mitgeteilt, ob der Steuerknüppel bewegt wird. Ist dies nicht der Fall, hat die Variable STICK0, die in Zeile 50 definiert wurde, den Wert von 15. Die Programmausführung wird mit Zeile 140 fortgesetzt.

ZEILE 230/250: wird der Steuerknüppel nach links bewegt (11), wird die Variable X um eins erniedrigt und die neue Position des Players in die Adresse D000 gePOKEt.

ZEILE 240/250: wird der Steuerknüppel nach rechts bewegt (7), wird die Variable X um eins erhöht und die neue Position des Players in die Adresse D000 gePOKEt.

ZEILE 260: wird der Steuerknüppel nach oben bewegt (14) dann wird jedes der 8 Byte des Player um ein Byte im Speicherbereich zurückgesetzt (Zeile 350). Dann wird die Variable Y um eins erniedrigt (Zeile 360).

ZEILE 270: wird der Steuerknüppel nach unten bewegt (13) dann wird jedes der 8 Byte des Player um ein Byte im Speicherbereich vorgerückt (Zeile 390). Dann wird die Variable Y um eins erhöht (Zeile 400).

ZEILE 290-320: wird der Steuerknüppel diagonal bewegt, erfolgt erst eine horizontale Bewegung und dann die vertikale. Dabei werden die Variablen X und Y auf den aktuellen Wert gesetzt.

ATARINSIDE

# ANHANG A
# BEISPIELPROGRAMME

Benutzte Fähigkeiten:
- vom Benutzer aufrufbare CIO-Routine (CIOVSR)(vgl.Anhang L)
- ganze Zahlen
- VARPTR-Anweisung
- ON ERROR-Anweisung

## DISKETTEN-DATEIEN-PROGRAMM

```
10 !                              ROUTINE ZUM LESEN
20 !                              DES DISKETTEN-INHALTSVERZEICHNISSES
30 !
40 ON ERROR 350
50 OPTION RESERVE(200)            !RESERVIERT SPEICHER FUER MASCHINENROUTINE
60 OPEN #1,"D:CIOUSR" INPUT
80 ADDR=VARPTR(RESERVE)           !BESTIMMT STARTADRESSE DES SPEICHERPLATZES
90 FOR I=0 TO 159                 !POKE DIE CIOUSR ROUTINE EIN
100 GET #1,D:POKE ADDR+I,D
110 NEXT I
120 CLOSE #1
130 PUTIOCB=ADDR                  !UMBENENNEN DER STARTADRESSE
140 CALLCIO=ADDR+61               !FUER JEDE DER ROUTINEN
150 GETIOCB=ADDR+81
160 DIM IOCB%(10)                 !DATEN DER ROUTINE = 10 BYTE
170 IOCB%(0)=1                    !BENUTZE UEBERTRAGUNGKANAL 1
180 IOCB%(1)=3                    !OEFFNE DISKETTENDATEI
190 IOCB%(5)=6                    !ZUM LESEN
200 FSPEC$="D:*.*"                !DATEI SPEZIFIKATION
210 !
220 Z=VARPTR(FSPEC$)              !ADRESSE DES STRINGS FSPEC$
230 Y=VARPTR(IOCB%(3))            !ADRESSE DES VARIABLENFELDES
240 POKE Y,PEEK(Z+2)              !HIGH BYTE DER ADRESSE
250 POKE Y+1,PEEK(Z+1)            !LOW BYTE DER ADRESSE
260 !
270 Z=USR(PUTIOCB,VARPTR(IOCB%(0)))
280 !
290 Z=USR(CALLCIO,VARPTR(IOCB%(0)))
300 !                             ÜBERTRAGUNGSKANAL IST GEOEFFNET
310 !                             DAS DISKETTEN-INHALTSVERZEICHNIS WIRD GELESEN
```



```
320 INPUT #1,S$
330 PRINT S$
340 GOTO 320
350 CLOSE #1
360 END
```

## EXPLOSIONS-UNTERPROGRAMM (SOUNDBEISPIEL)

```
10 !ZWEI ZEILEN HAUPTPROGRAMM
20 !UND EIN UNTERPROGRAMM ZUR
30 !ERZEUGUNG DES EXPLOSIONSGERAEUSCHES
40 !
50 GOSUB 8000
60 STOP
8000 !
8010 !EXPLOSIONS UNTERPROGRAMM
8020 SOUND 2,75,8,14
8040 ICR=0.79
8050 V1=15:V2=15:V3=15
8060 SOUND 0,NTE,8,V1
8070 SOUND 1,NTE+20,8,V2
8080 SOUND 2,NTE+50,8,V3
8090 V1=V1*ICR
8100 V2=V2*(ICR+.05)
8110 V3=V3*(ICR+.08)
8120 IF V3>1 THEN8060
8130 SOUND 0,0,0,0,0
8140 SOUND 1,0,0,0,0
8150 SOUND 2,0,0,0,0
8160 RETURN
```

## BEISPIEL FÜR FANFARENMUSIK

```
10 !ROUTINE ZUR ERZEUGUNG VON
20 !FANFARENMUSIK
30 !ZWEI ZEILEN HAUPTPROGRAMM
35 !
40 GOSUB 8000
50 STOP
8000 !
8010 !FANFAREN MUSIK
8020 !
8030 DUR=20:V0=181:V1=144:V2=121:GOSUB 8200
8040 DUR=7:GOSUB 8200
8050 GOSUB 8200
8060 DUR=9:V0=162:V1=128:V2=108:GOSUB 8200
8070 DUR=15:V0=181:V1=144:V2=121:GOSUB 8200
8080 V0=162:V1=128:V2=108:GOSUB 8200
8090 V0=153:V1=128:V2=96:V3=193
8100 FOR I=2 TO 14
8110 SOUND 3,V0,V1,10,I
```

ATARINSIDE

```
8120 SOUND 1,V1,10,I
8130 SOUND 2,V2,10,I
8140 SOUND 0,V3,10,I
8150 FOR J=1 TO 100:NEXT J
8160 NEXT I
8170 FOR J=1 TO 200:NEXT J
8180 SOUND 0,0,0,0,0
8185 SOUND 1,0,0,0,0
8190 SOUND 2,0,0,0,0
8195 SOUND 3,0,0,0,0
8197 RETURN
8200 ! SOUND-GENERATOR
8210 SOUND 0,V0,10,8,DUR
8220 SOUND 1,V1,10,8,DUR
8230 SOUND 2,V2,10,8,DUR
8240 !
8250 !ABBRECHEN DER TOENE
8260 !
8270 SOUND 0,0,0,0,0
8280 SOUND 1,0,0,0,0
8290 SOUND 2,0,0,0,0
8295 FOR J=1 TO 250:NEXT J
8300 RETURN
```

## DAS "ATARI-PIANO"

Benutzte Fähigkeiten:
- OPEN-Anweisung
- INKEY$-Anweisung
- SOUND-Anweisung
- Felder

```
10 !BEISPIELPROGRAMM MIT
20 !DEM DER ATARI-COMPUTER
30 !IN EIN PIANO VERWANDELT
35 !WIRD
40 !
50 !
60 !ZUNAECHST WERDEN DEN TASTEN
70 !DES COMPUTERS NOTENWERTE IM
80 !BEREICH VON 2 OKTAVEN ZUGEWIESEN
90 DIM NOTEN$(15)
100 DIM TON(15)
110 !DIESE WERDEN NUN ZUNAECHST
120 !IN DAS VARIABLENFELD EINGELESEN
130 OPEN #1,"D:NOTEN.DAT" INPUT
140 FOR I=1 TO 15
150 INPUT #1,S$,P
160 NOTEN$(I)=S$:TON(I)=P
170 NEXT I
180 CLOSE #1
190 PRINT "SPIELEN SIE BITTE !"
200 !
210 ! TESTBEGINN FUER TASTATUR
```



```
230 !
240 N$=INKEY$
250 IF N$="" THEN GOTO 240 ELSE GOTO 320
260 !
270 !WENN EINE TASTE GEDRUECKT IST
280 !WERDEN SIE ERKENNEN OB ES EINE
290 !DES PIANOS IST.
300 !
310 !
320 FOR I=1 TO 15
330 IF N$=NOTEN$(I) THEN 380
340 NEXT I
350 GOTO 240 !FALSCHE TASTE. NEUER VERSUCH
360 !
380 VOLUME=8
390 SOUND 1,TON(I),10,VOLUME,15
400 GOTO 240
410 END
```

```
10 !PROGRAMM ZUM ERSTELLEN
20 !DER DATEI NOTEN.DAT
30 DIM NOTEN$(15),TON(15)
40 FOR I=1 TO 15
50 INPUT "BITTE TASTE DRUECKEN, FREQ. ANGEBEN:";NOTEN$(I),TON(I)
60 NEXT I
70 OPEN #1,"D:NOTEN.DAT" OUTPUT
80 FOR I=1 TO 15
90 PRINT #1,NOTEN$(I);",";TON(I)
100 NEXT I
110 CLOSE #1
120 END
```

Um eine zwei Oktaven-Tastatur zu erhalten geben Sie folgende Werte ein:

```
Z, 243
X, 216
C, 193
V, 182
B, 162
N, 144
M, 128
A, 121
S, 108
D, 96
F, 91
G, 81
H, 72
J, 64
K, 60
```

ATARINSIDE

## DEZIMAL - HEXADEZIMALUMWANDLUNG

Benutzte Fähigkeiten:
- Felder
- ganze Zahlen

```
 20 !
 30 !DEZI HEXA
 40 !
 50 !
 60 !
 70 ! PROGRAMM ZUM UMWANDELN
 80 ! EINER DEZIMALZAHL IN
 90 ! DIE ENTSPRECHENDE HEXADEZIMALZAHL
 100 !
 110 !
 130 DIM HEX$(15):DIM HEXBASE(4)
 140 FOR I=0 TO 15
 150 READ HEX$(I)
 160 NEXT I
 170 FOR I=0 TO 4
 180 READ HEXBASE(I)
 190 NEXT I
 200 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
 210 DATA 0,4096,256,16,1
 220 !
 230 !EINGEBEN DER DEZIMALZAHL
 240 !
 250 INPUT "GEBEN SIE DIE DEZIMALZAHL EIN ";DEZ
 260 IF DEZ=0 THEN 500 !STOP
 270 !
 280 !UMRECHNUNG
 290 !
 300 FOR J=1 TO 4
 305 IF J=4 THEN ANS%=DEZ:GOTO 350
 310 ANS% = (DEZ/HEXBASE(J))-.5
 320 IF ANS% < 1 THEN ANS% = 0
 330 DEZ = DEZ-(ANS%*HEXBASE(J))
 340 !
 350 !
 360 FOR I% = 0 TO 15
 370 IF ANS% = I% THEN 420
 380 NEXT I%
 390 ! FEHLER !
 400 PRINT "ZAHL KANN NICHT UMGEWANDELT WERDEN."
 410 PRINT "BITTE NEUE ZAHL EINGEBEN!";GOTO 250
 420 HEXNO$ = HEXNO$+HEX$(I%)
 430 NEXT J
 440 !
 450 !DIE HEXZAHL WIRD AUSGEGEBEN
 460 !
 470 PRINT "HEXZAHL = ";HEXNO$
 480 HEXNO$ = "":GOTO 250
 500 END
```



## VERTIKALES FEINVERSCHIEBEN DES BILDSCHIRMINHALTS

Benutzte Fähigkeiten:
- VARPTR-Anweisung
- OPTION RESERVE und CHR1
- Display List

```
 10 DEFINT A-Z
 20 OPTION RESERVE(3000)                !SPEICHERPLATZ FUER BILDSCHIRM
 30 OPTION CHR1                         !SPEICHERPLATZ FUER DISPLAY LIST
 40 ADDR=VARPTR(CHR1)
 50 CADDR=VARPTR(RESERVE)
 60 VSCROL=&D405                        !ADRESSE VERTIKALES SCROLLING
 70 LCADDR=0
 80 HCADDR=((CADDR AND &FF00)/256) AND &FF
 90 FOR I=0 TO 99                       !SPEICHERPLATZ (ZEILE 30) LOESCHEN
 100 POKE ADDR+I,0:NEXT I
 110 LADDR=ADDR AND &FF
 120 HADDR=((ADDR AND &FF00)/256) AND &FF
 130 LMSLO=ADDR+4                       !ADRESSE ZUM LADEN
 140 LMSHI=ADDR+5                       !DER SPEICHERBYTE (LMS)
 150 FOR I=0 TO 18                      !POKE IN NEUE DISPLAY LIST
 160 READ D
 170 POKE ADDR+I,D
 180 NEXT I
 190 DATA &70,&70,&70,&67,&00,&00,&27,$27
 200 DATA &27,&27,&27,&27,&27,&27,&27,&27
 210 DATA &27,&07,&41
 220 POKE ADDR+19,LADDR                 !STARTADRESSE DER NEUEN
 230 POKE ADDR+20,HADDR                 !DISPLAY LIST
 240 POKE LMSLO,LCADDR:POKE LMSHI,HCADDR  ! BILDSCHIRMSPEICHER
 250 K=-1
 260 FOR I=1 TO 300
 270 K=K+1:POKE CADDR+K,33
 280 NEXT I
 290 FOR I=34 TO 58
 300 FOR J=1 TO 20
 310 K=K+1:POKE CADDR+K,I
 320 NEXT J,I
 330 POKE &22F,0                        !ANTIC AUSSCHALTEN
 340 POKE &230,LADDR                    !MITTEILEN, WO SICH
 350 POKE &231,HADDR                    !DISPLAY LIST BEFINDET
 360 POKE &22F,&22                      !ANTIC ANSCHALTEN
 370                                    !HIER BEGINNT DAS SCROLLEN
 380 FOR I=1 TO 15
 390 POKE VSCROL,I
 400 FOR W=1 TO 30:NEXT W
 410 NEXT I
```

ATARINSIDE

```
420 CADDR=CADDR+20                    !BERECHNET, WO DIE NAECHSTE ZEILE
430 LADDR=CADDR AND &FF               !DES BILDSCHIRM-SPEICHERS STARTET
440 HCADDR=((CADDR AND &FF00)/256)AND&FF
450 WAIT &D40B,&FF,96                 !WARTET BIS TV ZEILENZAEHLER 96 IST
460 POKE VSCROL,0
470 POKE LMSLO,LCADDR
480 POKE LMSHI,HCADDR                 !SCROLLING BEGINNT
490 GOTO 380
500 END
```



# BEISPIELPROGRAMME GRAFIK  ANHANG B

## MIKROBENANGRIFF

```
 10 REM MIKROBEN INVASION
 20 REM SPIRALKREATUREN BEWEGEN
 25 REM SICH UEBER DEN BILDSCHIRM
 30 RANDOMIZE
 40 MODE=RND(8)
 50 GRAPHICS MODE+16
 60 PIX=RND(15)
 70 SETCOLOR 0,PIX,6
 80 COLOR 1
 90 BAK=RND(255)
 100 POKE 712,BAK
 110 X=RND(150):Y=RND(100)
 120 IF X>140 THEN 40
 130 Z=2
 140 NUM=NUM+1
 150 FOR DOTS=1 TO Z
 160 IF NUM=5 THEN NUM=1
 170 ON ERROR GOTO 230
 180 PLOT X,Y
 190 ON NUM GOSUB 250,270,290,310
 200 NEXT
 210 Z=Z+1
 220 GOTO 140
 230 GRAPHICS MODE+32+16          !KEIN TEXTFENSTER,KEIN LOESCHEN
 240 RESUME 60
 250 X=X+1:Y=Y+1
 260 RETURN
 270 X=X+1:Y=Y-1
 280 RETURN
 290 X=X-1:Y=Y-1
 310 X=X-1:Y=Y+1
 320 RETURN
 330 END
```

ATARINSIDE

TOP SECRET-PROGRAMM

Das nächste Programm zeigt die Verwendung der Anweisungen RANDOMIZE und RND und erzeugt Wörter und Abkürzungen mit drei Buchstaben.

```
10 RANDOMIZE                    !SETZT RANDOM-FUNKTION
20 GRAPHICS 2+16
30 X=RND(26)+96                 !ERZEUGT ERSTEN BUCHSTABEN
40 Y=RND(5)                     !ERZEUGT EINEN VOKAL FUER MITTLEREN BUCHSTABEN
50 IF Y=1 THEN Y=97             !ERZEUGT EIN A
60 IF Y=2 THEN Y=101            !ERZEUGT EIN E
70 IF Y=3 THEN Y=105            !ERZEUGT EIN I
80 IF Y=4 THEN Y=111            !ERZEUGT EIN O
90 IF Y=5 THEN Y=117            !ERZEUGT EIN U
100 Z=RND(26)+96                !ERZEUGT LETZTEN BUCHSTABEN
110 PRINT #6,AT(9,3)CHR$(X);CHR$(Y);CHR$(Z)
120 FOR ZEIT=1 TO 2000:NEXT
180 GOTO 30
```



# VERANDERN DES ZEICHENSATZES

# ANHANG C

Die ATARI-Computer haben einen Zeichensatz (Character-Set), der im ROM (Read only Memory / Nur-Lese-Speicher) festgelegt ist. Dieser enthält die Groß- und Kleinbuchstaben, Ziffern, Spezialzeichen und den Grafikzeichensatz (CTRL-Zeichen). Es ist möglich, einen eigenen Zeichensatz zu erstellen, sei es nun um internationale Zeichen zu erhalten (Aüß u.ä.) oder um Spielfiguren und Hintergründe für Spiele zu entwerfen.

Mit dem ATARI Microsoft BASIC II wird die Erstellung des eigenen Charaktersatzes erleichtert. Das Operations System (OS) des ATARI liest vor jeder Ausgabe eines Zeichens erst in der Adresse 756 (hex 2F4) nach, wo sich der Zeichensatz befindet. Diese Adresse (POINTER) wird daher CHBAS (CHarakter BASis genannt). Normalerweise zeigt dieser POINTER auf den Zeichensatz im ROM-Bereich.

Mit dem BASIC-Befehl POKE kann ein anderer Zeichensatz erzeugt werden. Dieser kann nur im RAM-Bereich angelegt werden (RAM ist der dem Anwender zugängliche Speicherbereich des Computers). Dazu kann im RAM-Bereich mit den Anweisungen OPTION CHR1 oder OPTION CHR2 ein bestimmter Speicherbereich reserviert werden. Dann wird dem oben genannten POINTER (CHBAS) ein neuer Wert, die Startadresse des neuen Zeichensatzes, zugewiesen.

Beachten Sie folgende Hinweise bei der Erstellung eines eigenen Zeichensatzes:

1. der Grafikmodus 0 erlaubt die Darstellung von 128 Zeichen. Um den Speicherplatz zu reservieren benutzen Sie OPTION CHR1. Im Grafikmodus 1 und 2 können 64 Zeichen dargestellt werden. Hier lautet die Anweisung zum Reservieren OPTION CHR2.

2. Um den Zeichensatz verändern zu können, muß der Originalzeichensatz erst einmal in den reservierten Speicherbereich kopiert werden. Dadurch ist es möglich, nur einige Zeichen zu verändern und den Rest der Originalzeichen zu übernehmen.

3. Der 64 Zeichen Satz benötigt 512 Byte Speicherplatz (8 Bytes pro Zeichen). Bei der Verwendung des 128-Zeichen-Satzes werden demzufolge 1024 Byte Speicherplatz benötigt. Dieser wird, wie schon unter Punkt 1. aufgeführt, durch die OPTION-Anweisung reserviert.

4. Der Wert, der in die Adresse CHBAS gePOKEt werden muß, läßt sich mit der folgenden Anweisung errechnen: CHBAS%=(VARPTR(CHRn)/256) AND 3FF. Dabei ist n 1 oder 2, entsprechend der OPTION-Anweisung CHR1 oder CHR2. Der so ermittelte Wert wird in die Adresse 756 (hex 2F4) gePOKEt.

ATARINSIDE

Die meiste Zeit bei der Erstellung eines eigenen Zeichensatzes wird bei der Entwicklung und Auswertung der einzelnen Zeichen benötigt. Jedes Zeichen besteht aus 8 Bytes, dabei liegt das erste Byte oben. Da ein Byte aus acht bit besteht (bit ist die kleinste Einheit in der Computertechnik), ergibt sich das Zeichen aus einer 8 mal 8 Matrix. Um ein Zeichen zu entwickeln, gehen Sie ähnlich vor, wie bei der Erstellung eines Player-Objektes. Nehmen Sie ein Blatt kariertes Papier. Teilen Sie davon durch Einkreisen 8 mal 8 Kästchen ab. Dann malen Sie die Kästchen aus, bei denen im Zeichen ein Punkt erscheinen soll. (siehe Abb. C.2).

Dann wird jede waagrechte Reihe in eine Binärzahl umgewandelt. Dabei entspricht ein ausgemaltes Kästchen einer 1 und ein freies einer 0. Die so ermittelten Binärzahlen werden dann in Hexzahlen oder Dezimalzahlen umgewandelt. (siehe Tabelle 2.1, Seite 14).

Diese Zahlen werden dann in den reservierten Speicherbereich gePOKET. Dabei repräsentieren die ersten acht Byte ein leeres Zeichen (Zeichen 32 in ANLAGE K), die nächsten 8 das erste Zeichen usw. Nach dem Kopieren des Originalzeichensatzes in den reservierten Speicherbereich können Sie die Position des zu verändernden Buchstaben ermitteln. Dazu ein Beispiel:

Die Ausgabe des neuen Zeichens geschieht durch PRINT "A".

Abb C.1 Verteilung der Bytes eines Zeichens

BYTE 1 (bit 1-8)
BYTE 2 (bit 9-16)
BYTE 3 (bit 17-24)
BYTE 4 (bit 25-32)
BYTE 5 (bit 33-40)
BYTE 6 (bit 41-48)
BYTE 7 (bit 49-56)
BYTE 8 (bit 57-64)



| Byte Nummer | | Binärzahl | Hexzahl | Dezimalzahl |
|---|---|---|---|---|
| 1 | | 00110000 = | 30 = | 48 |
| 2 | | 00110000 = | 30 = | 48 |
| 3 | | 11111000 = | F8 = | 248 |
| 4 | | 00011100 = | 1C = | 28 |
| 5 | | 00001110 = | 0E = | 14 |
| 6 | | 00000111 = | 07 = | 07 |
| 7 | | 00000011 = | 03 = | 03 |
| 8 | | 00000011 = | 03 = | 03 |

Abbildung C-2 Ändern eines Zeichens

BEISPIELPROGRAMM:

```
10 !
20 !                                PROGRAMM ZUM ERSTELLEN UND
30 !                                AUSGEBEN EINES ZEICHENS
40 !
60 !                                DIESES PROGRAMM WANDELT DIE
70 !                                ZEICHEN A,B,C,D,E,F,G UND H UM
80 !
90 CHBAS=&2F4                      !POINTER DES ZEICHENSATZES
100 OPTION CHR1                    !RESERVIERT SPEICHERPLATZ
110 ADDR%=VARPTR(CHR1)             !ERMITTELT STARTADRESSE
120 PAGENO%=(ADDR%/256)AND &FF     !ERMITTELT PAGE(SEITE)
130 !
140 MOVE 57344,ADDR%,1024          !KOPIERT ZEICHENSATZ IN RAM BEREICH
150 !
160 OFFSET=33*8                    !ERMITTELN POS. BUCHSTABE A
170 FOR I=0 TO 63                  !ERSTELLEN NEUER ZEICHEN
180 READ C
190 POKE ADDR%+OFFSET+I,C
200 NEXT
210 !
220 !                               DATA ZEILEN DER NEUEN ZEICHEN
230 !
240 DATA &07,&0F,&1F,&3F,&7F,&FF,&FF,&FF
250 DATA &E0,&F0,&FB,&FC,&FE,&FF,&FF,&FF
260 DATA &FF,&FF,&FF,&7F,&3F,&1F,&0F,&07
270 DATA &FF,&FF,&FF,&FE,&FC,&FB,&F0,&E0
280 DATA &00,&00,&00,&EF,&7F,&FF,&FF,&FF
290 DATA &00,&00,&00,&FC,&FE,&FF,&FF,&FF
300 DATA &FF,&FF,&FF,&7F,&3F,&00,&00,&00
310 DATA &FF,&FF,&FF,&FE,&FC,&00,&00,&00
320 !
330 POKE CHBAS,PAGENO%            !UMSCHALTEN AUF NEUEN ZEICHENSATZ
```

ATARINSIDE

```
350 POKE &2F0,1                    !SCHALTET CURSOR AUS
360 SETCOLOR 6,2,6
370 X=20
380 FOR Y=10 TO 20
390 WAIT &D40B,&FF,110
400 CLS
410 PRINT AT(X,Y+1);"CD"
420 FOR W=1 TO 30:NEXT W
430 NEXT Y
440 CLS
450 PRINT AT(X,22);"GH"
460 SOUND 0,79,10,8,4
470 FOR W=1 TO 80:NEXT W
480 FOR Y=20 TO 10 STEP-1
490 WAIT &D40B,&FF,110
500 CLS
510 PRINT AT(X,Y+1);"CD"
520 FOR W=1 TO 30:NEXT W
530 NEXT Y
540 GOTO 380
```



# EINGABE-AUSGABEGERÄTE

# ANHANG D

Die Tastatur, das Diskettenlaufwerk, der Programm-Rekorder und das Modem geben dem Computer Informationen (Daten). Es wird ein Eingabegerät benutzt.

Der Computer gibt Daten an den Bildschirm, das Diskettenlaufwerk, den Programm-Rekorder und den Drucker. Dazu wird ein Ausgabegerät benutzt.

Die Geräte werden durch die Eingabe eines Buchstabens zur Übertragung von Daten angesprochen.
Es bedeuten:

K: steht für KEYBOARD (Tastatur des Computers). Dieser Kanal ist nur zum Lesen geeignet.

P: steht für PRINTER (Drucker). Dieser Kanal ist nur zum Schreiben von Daten bestimmt.

C: steht für CASSETTE (Programm-Rekorder). Dieser Kanal kann zum Lesen und Schreiben von Daten geöffnet werden. Allerdings kann immer nur eine Funktion zur gleichen Zeit stattfinden. Gleichzeitiges Lesen und Schreiben von Daten ist nicht möglich. Der Programm-Rekorder arbeitet im Vier-Spur-Verfahren (Stereo). Dadurch werden auf eine Spur die Daten aufgezeichnet und die andere Spur steht für Musik oder Kommentare zur Verfügung. Lesen Sie bitte dazu auch das Bedienungshandbuch Ihres Programm-Rekorders.

D1:, D2:, D3:, D4: steht für das jeweilige DISK DRIVE (Diskettenlaufwerk 1 bis 4). Auch hier ist das Lesen und Schreiben von Daten möglich. Ohne Ziffernangabe entspricht D: immer dem Laufwerk D1:.

E: steht für SCREEN EDITOR (Bildschirm/Tastatur). Es ist eine Eingabe (schreiben) und eine Ausgabe (lesen) von Daten möglich. Dieser Datenkanal verwandelt Computer und Bildschirm in ein Terminal (nur Grafikmode 0). Alle Daten die eingegeben werden, erscheinen auf dem Bildschirm an der jeweiligen Cursorposition. Das Lesen von Daten wird durch Drücken der RETURN-Taste ausgelöst. Die Daten der letzten Zeile werden vom Programm übernommen.

S: steht für SCREEN (Bildschirm). Dieser Datenkanal erlaubt das Schreiben und Lesen vom Bildschirm und das Positionieren des Cursors. Außerdem können Grafikoperationen durchgeführt werden.

R: steht für RS-232 (Schnittstelle, Interface). Hier wird das Interface-Modul angesprochen. Es verbindet den ATARI-Computer mit solchen Peripheriegeräten, die diese weit verbreitete Schnittstelle benötigen (Plotter, Drucker, Terminal).

ATARINSIDE

# ANHANG E

## WAS LIEGT WO IM SPEICHER ?

Nachfolgend sind die wichtigsten Adressen im Speicher des ATARI-Computers aufgeführt. Die Angabe der Adresse erfolgt erst hexadezimal und dann dezimal. Weiterhin wird der (amerikanische) Name der Adresse angegeben.

### SPEICHERMAPPE

Der im ATARI verwendete Mikroprozessor 6202 ist aufgeteilt in vier Haupt-Speicherbereiche. Den RAM- und Einsteckmodulbereich, den Eingabe- / Ausgabebereich und den ROM-Bereich. Aus der folgenden Aufstellung können Sie ersehen wo jeder dieser vier Bereiche beginnt und endet:

| | |
|---|---|
| RAM (minimale Ausbaustufe) | :0000-3FFF (0-16383) |
| RAM (Ausbaustufen) | :4000-7FFF (16384-32767) |
| Steckmodul B (linkes Modul) oder 8k RAM | :8000-9FFF (32768-40959) |
| Steckmodul A (rechtes Modul) oder 8k RAM | :A000-BFFF (40960-49151) |
| Unbenutzter Bereich | :C000-CFFF (49152-53247) |
| Eingabe- / Ausgabebereich (I/O-Chips) | :D000-D7FF (53248-55295) |
| Fließkommaarithmetik im ROM | :D800-DFFF (55296-57343) |
| Operating-System-ROM-Bereich | :E000-FFFF (57344-65535) |

### RAM-BEREICH

Der RAM-Bereich, also der dem Anwender zur Verfügung stehende Speicherplatz, enthält einige Adressen, die vom Operating-System des Computers benutzt werden. Der RAM-Bereich unterteilt sich daher in folgende Unterbereiche:

. PAGE 0 (Seite 0). Hier liegen die Zeiger (Pointer) des Operating-Systems. (Uhren, Variablenzeiger usw.). Sie liegen im Bereich von 0000 bis 00FF (0-255).
0000 bis 007F (0-127) : Operating-System.
0080 bis 00FF (128-255): steht dem Anwender zur Verfügung.
00D4 bis 00FF (212-255): wird von den Fließkommaroutinen benutzt.

. PAGE 1 Hier liegt der STACK (Stapel) des Computers. Er hat die Adressen von 0100 bis 01FF (256-511).

ANMERKUNG: Diese Stacks werden bei jedem Einschalten des Computers oder bei einem SYSTEM-RESET initialisiert.

. PAGE 2-4. Hier sind die Variablen, Tabellen und Datenbuffer abgelegt. Der Bereich wird OS-DATA-BASE genannt und umfaßt die Adressen von 0200 bis 047F (512-1151).

. PAGE 7-XX. Dieser Bereich ab 0700 (1792) steht dem Anwender zur Verfügung. Der Wert XX hängt von verwendeten Grafikmodus und dem vorhandenen Speicher ab.

. Seite XX bis zum Ende des RAM-Bereichs. Hier liegen die Adressen der Display List und der Bildschirmdaten. Der DATA-PAGE-POINTER enthält dabei jeweils die unterste Adresse des Bildschirmspeichers.



### BEREICH DER STECKMODULE (CARTRIDGE)

Einsteckmodul B ist das rechte beim ATARI 800 Privat-Computer. Steckmodul A ist demzufolge das linke und das einzige bei allen anderen ATARI-Computern.
Die Bereiche der Einsteckmodule im Speicherbereich:
. Einsteckmodul B : 8000 bis 9FFF (32768-40959)
. Einsteckmodul A : A000 bis BFFF (40960-49151) bei 8k-Modulen
. Einsteckmodul A : 8000 bis BFFF (32768-49151) bei 16k-Modulen

ANMERKUNG: Wenn ein RAM-Modul (Speichererweiterung) im letzten Steckplatz des Computers installiert ist (nur ATARI 800), benutzt es einen Teil des Speicherbereichs für das Steckmodul A. Daher werden bei Benutzung dieses Moduls 8k Byte abgezogen.

### EINGABE- / AUSGABEBEREICH (I/O CHIPS)

Der Mikroprozessor benutzt bei allen Ein- und Ausgabeoperationen einen der vorhandenen Spezialbausteine. Deren Adressen im Speicherbereich sind nachfolgend aufgeführt:
. GTIA/CTIA : D000 bis D01F (53248-53279)
. POKEY : D200 bis D21F (53760-53791)
. PIA : D300 bis D31F (54016-54047)
. ANTIC : D400 bis D41F (54272-54303)

Ein Teil dieser Chips hat nur die Möglichkeit Daten zu lesen, andere wiederum können nur Daten schreiben. In der Tabelle E.2 sind die Chips und ihre Adressen im Speicher nochmals angegeben. Für weitere Informationen schauen Sie bitte in den technischen Unterlagen des ATARI-Computers nach.

### ROM-Bereich (Operating-System)

Der Bereich von D800 bis FFFF (55296-65535) wird ständig vom Operating-System des Computers benutzt. Hier liegen z.B. die Unterroutinen zur Bildschirmausgabe, der Zeichensatz und nicht zuletzt die Fließkommaarithmetik (Gleitkomma-Arithmetik).
. Gleitkommaroutinen (Floating point): D800 bis DFFF (55296-57343)
. Operating-System-ROM (OS-ROM) : E000 bis FFFF (57344-65535)

Tabelle E.1 Adressen der OS-DATA-BASE

| ADRESSE HEX | ADRESSE DEZIMAL | NAME | BYTE ANZAHL | FUNKTION |
|---|---|---|---|---|
| SPEICHERANGABEN | | | | |
| 000E | 14 | APPMHI | 2 | höchste Adresse des vom Anwender nutzbaren Speicherbereichs |
| 006A | 106 | RAMTOP | 1 | höchste Adresse des RAM-Bereichs (wird vom Display-Handler benutzt!) |
| 02E4 | 740 | RAMSIZ | 1 | höchste Adresse des RAM-Bereichs |
| 02E5 | 741 | MEMTOP | 2 | Zeiger zur höchsten Adresse des Anwenderspeicherbereichs |
| 02E7 | 742 | MEMLO | 2 | Zeiger zur niedrigsten Adresse des Anwenderspeicherbereichs |

ATARIINSIDE

TEXT- UND GRAFIKBILDSCHIRM

| | | | | |
|---|---|---|---|---|
| 0052 | 82 | LMARGN | 1 | linker Rand des Bildschirms. |
| 0053 | 83 | RMARGN | 1 | rechter Rand des Bildschirms. |

Cursor-Kontrolle

| | | | | |
|---|---|---|---|---|
| 0054 | 84 | ROWCRS | 1 | Zeile, in der sich der Cursor befindet. |
| 0055 | 85 | COLCRS | 2 | Spalte in der sich der Cursor befindet. |
| 005A | 90 | OLDROW | 1 | vorherige Zeile des Cursors. |
| 005B | 91 | OLDCOL | 2 | vorherige Spalte des Cursors. |
| 0290 | 656 | TXTROW | 1 | Zeile des Textfensters, in der sich der Cursor gerade befindet. |
| 0291 | 657 | TXTCOL | 2 | Spalte des Textfensters, in der sich der Cursor gerade befindet. |
| 02F0 | 752 | CRSINH | 1 | schaltet Cursor an oder aus. 0 schaltet ein, 1 schaltet aus. |

Farben-Kontrolle

| | | | | |
|---|---|---|---|---|
| 02C0 | 704 | PCOLR0 | 4 | Farbe von Player/Missile 0 |
| 02C1 | 705 | PCOLR1 | 4 | Farbe von Player/Missile 1 |
| 02C2 | 706 | PCOLR2 | 4 | Farbe von Player/Missile 2 |
| 02C3 | 707 | PCOLR3 | 4 | Farbe von Player/Missile 3 |
| 02C4 | 708 | COLOR0 | 5 | Farbe vom Spielfeld 0 |
| 02C5 | 709 | COLOR1 | 5 | Farbe vom Spielfeld 1 |
| 02C6 | 710 | COLOR2 | 5 | Farbe vom Spielfeld 2 |
| 02C7 | 711 | COLOR3 | 5 | Farbe vom Spielfeld 3 |
| 02C8 | 712 | COLOR4 | 5 | Farbe vom Hintergrund |

Farbumschaltung zum Schutz der Bildröhre (Attract-Mode)

| | | | | |
|---|---|---|---|---|
| 004D | 77 | ATRACT | 1 | schaltet nach ca. 7 Minuten in den Attract Mode. Wert 128 schaltet ein. |

Tabulator-Adressen

| | | | | |
|---|---|---|---|---|
| 02A3 | 675 | TABMAP | 15 | Tabelle der Tabulatorpositionen. |

Bildschirmspeicher

| | | | | |
|---|---|---|---|---|
| 0058 | 88 | SAVMSC | 2 | obere linke Ecke des Bildschirms. |

Grafikbildschirm mit Textfenster

| | | | | |
|---|---|---|---|---|
| 0294 | 660 | TXTMSC | 2 | obere linke Ecke des Textfensters. |



Fill-Funktion

| | | | | |
|---|---|---|---|---|
| 02FD | 765 | FILDAT | 1 | Daten für den FILL-Befehl. |

Interner Code des Zeichensatzes

| | | | | |
|---|---|---|---|---|
| 02FA | 762 | ATACHR | 1 | Wert des letzten Zeichens oder PLOT-Punktes |

Grafikzeichensatz- (CTRL-Zeichen) Kennung

| | | | | |
|---|---|---|---|---|
| 02FE | 766 | DSPFLG | 1 | enthält Wert 1 bei Benutzung von CTRL-Zeichen, (sonst 0) |

TASTATUR (Keyboard)

Tastenwert lesen

| | | | | |
|---|---|---|---|---|
| 02FC | 764 | CH | 1 | enthält den Wert der zuletzt gedrückten Taste. Zum Löschen $FF (255) POKEn. |

Spezialfunktionen

| | | | | |
|---|---|---|---|---|
| 0011 | 17 | BRKKEY | 1 | steht in dieser Adresse eine 0 wurde die BREAK-Taste gedrückt. |
| 02B6 | 694 | INVFLG | 1 | eine 128 gibt an, daß invers geschrieben wird (ATARI-Taste) Sonst ist der Wert 0. |
| 02BE | 702 | SHFLOK | 1 | 0 = Kleinbuchstaben, 64 = Großbuchstaben, 128 = Grafikzeichen. Durch SYSTEM RESET wird hier der Wert 64 gesetzt. |
| 02FF | 767 | SSFLAG | 1 | Start- / STOP-Zeiger. Er wird verändert durch CTRL 1. Normalwert ist 0. |

ATARINSIDE

EINGABE- / AUSGABEBEREICH (Central-I/O-Routine)

Eingabe- / Ausgabe-Kontroll-Blocks

| | | | | |
|---|---|---|---|---|
| 0340 - | 832 - | IOCB | 16 | E/A-Kontrollblock 0. |
| 034F | 847 | -- | -- | - |
| 0350 - | 848 - | IOCB | 16 | E/A-Kontrollblock 1. |
| 035F | 863 | -- | -- | - |
| 0360 - | 864 - | IOCB | 16 | E/A-Kontrollblock 2. |
| 036F | 879 | -- | -- | - |
| 0370 - | 880 - | IOCB | 16 | E/A-Kontrollblock 3. |
| 037F | 895 | -- | -- | - |
| 0380 - | 896 - | IOCB | 16 | E/A-Kontrollblock 4. |
| 038F | 911 | -- | -- | - |
| 0390 - | 912 - | IOCB | 16 | E/A-Kontrollblock 5. |
| 039F | 927 | -- | -- | - |
| 03A0 - | 928 - | IOCB | 16 | E/A-Kontrollblock 6. |
| 03AF | 943 | -- | -- | - |
| 03B0 - | 944 - | IOCB | 16 | E/A-Kontrollblock 7. |
| 03BF | 959 | -- | -- | - |
| 0340 | 832 | ICHID | 1 | Handler-Identifikation. Wird durch SYSTEM RESET initialisiert. |
| 0341 | 833 | ICDNO | 1 | Nummer des Datenkanals. |
| 0342 | 834 | ICCMD | 1 | Kommando-Byte. |
| 0343 | 835 | ICSTA | 1 | STATUS |
| 0344 | 836 | ICBAL/ICBAH | 2 | Puffer-Adresse (Low- /High-Byte). |
| 0346 | 838 | ICPTL/ICPTH | 2 | Zeiger für Datenausgabe. Weist nach SYSTEM-RESET auf geschlossenen Übertragungskanal. |
| 0348 | 840 | ICBLL/ICBLH | 2 | Puffer-Länge |
| 034A | 842 | ICAX1/ICAX2 | 2 | Hilfs-Byte |
| 034C | 844 | ICAX3/ICAX4 | 4 | Reserve-Byte. |

Eingabe- / Ausgabe-Kontollblocks der PAGE 0

| | | | | |
|---|---|---|---|---|
| 0020 | 32 | ZIOCB | 16 | E/A-Kontrollblock PAGE 0. |
| 0020 | 32 | ICHIDZ | 1 | Index für geöffnete Datei. Nach einem CLOSE steht hier $FF (255). |
| 0021 | 33 | ICDNOZ | 1 | Laufwerk Nummer (1-4). |
| 0022 | 34 | ICCOMZ | 1 | Kommando-Byte. |
| 0023 | 35 | ICSTAZ | 1 | STATUS |
| 0024 | 36 | ICBALZ/ICBALH | 2 | wie Adresse 0344 (836). |
| 0026 | 38 | ICPTLZ/ICPTHZ | 2 | wie Adresse 0346 (838). |
| 0028 | 40 | ICBLLZ/ICBLHZ | 2 | wie Adresse 0348 (840). |
| 002A | 42 | ICAX1Z/ICAX2Z | 2 | Hilfs-Byte. |
| 002C | 44 | ICSPRZ | 4 | gibt bei DISK-Operationen den Diskettensektor an. |

ZUSTAND DES GERATES (Device-Status)

| | | | | |
|---|---|---|---|---|
| 02EA | 746 | DVSTAT | 4 | Zustand des Gerätes. |

GERATE-TABELLE (Device-Table)

| | | | | |
|---|---|---|---|---|
| 031A | 749 | HATABS | 38 | Tabelle der benutzten Geräte. |



SERIELLE EIN- / AUSGABE-ROUTINEN (Serial-I/O-Routine)

Geräte-Kontrollbereich (Device-Control-Block)

| | | | | |
|---|---|---|---|---|
| 0300 - | 768 - | DCB | 12 | Geräte-Kontrollblock. |
| 030B | 779 | - | -- | - |
| 0300 | 768 | DDEVIC | 1 | Geräte-Identifikation. |
| 0301 | 769 | DUNIT | 1 | Disketten Laufwerk Nr. |
| 0302 | 770 | DCOMND | 1 | auszuführender Diskettenbefehl. |
| 0303 | 771 | DSTATS | 1 | Status |
| 0304 | 772 | DBUFLO/DBUFHI | 2 | Datenpuffer Low-/High-Byte. |
| 0306 | 774 | DTIMELO | 1 | Zeitbegrenzung für Handler. |
| 0308 | 776 | DBYTLO/DBYTHI | 2 | Puffer-Länge. |
| 030A | 778 | DAUX1,DAUX2 | 2 | Hilfs-Byte. |

KLANGERZEUGUNG (Bus-Sound-Control)

| | | | | |
|---|---|---|---|---|
| 0042 | 65 | SOUNDR | 1 | 0 schaltet Ton der Übertragung aus. Geräuschloses Laden und Speichern von Daten/Programmen. |

SPIELKONTROLLGERATE (ATARI-Controllers)

Steuerknüppel (Joysticks)

| | | | | |
|---|---|---|---|---|
| 0278 | 632 | STICK0-STICK3 | 4 | Richtung des Steuerknüppels. |
| 0284 | 644 | STRIG0-STRIG3 | 4 | Feuerknöpfe der Steuerknüppel. |

Drehregler (Paddles)

| | | | | |
|---|---|---|---|---|
| 0270 | 624 | PADDL0-PADDL7 | 8 | Stellung der Drehregler. |
| 027C | 636 | PTRIG0-PTRIG7 | 8 | Knöpfe der Drehregler. |

Lichtgriffel (Light-Pen)

| | | | | |
|---|---|---|---|---|
| 0234 | 564 | LPENH | 1 | Zeile des Lichtgriffels. |
| 0235 | 565 | LPENV | 1 | Spalte des Lichtgriffels. |
| 0278 | 632 | STICK0-STICK3 | 4 | Knopf des Lichtgriffels. |

GLEITKOMMA- / FLIESSKOMMABEREICH (Floating-Point-Package)

| | | | | |
|---|---|---|---|---|
| 00D4 | 212 | FR0 | 6 | Gleitkommaregister 0. |
| 00E0 | 224 | FR1 | 6 | Gleitkommaregister 1. |
| 00F2 | 242 | CIX | 1 | Zeichen Index. |
| 00F3 | 243 | INBUFF | 1 | Zeiger zum Texteingabepuffer. |
| 00FB | 251 | DEGFLG/RADFLG | 1 | hier steht, ob in Radiant (0) oder Grad (6) gerechnet wird. |
| 00FC | 252 | FLPTR | 2 | Zeiger zur Gleitkommastelle. |
| 0580 | 1408 | LBUFF | 96 | Text Puffer. |

ATARINSIDE

EINSCHALTEN DES COMPUTER UND SYSTEM-RESET (Power-up)

Disketten / Kassetten automatisch laden (Diskette/Casette-Boot)

| | | | | |
|---|---|---|---|---|
| 0002 | 2 | CASINI | 2 | Zeiger zur Kassetten-BOOT-initialisierungsadresse. |
| 000C | 12 | DOSINI | 2 | Zeiger zur DOS-Initialisierungsadresse. |

Kontrolle des Einsteckmoduls / des Warmstartzeigers (Enviroment-Control)

| | | | | |
|---|---|---|---|---|
| 0008 | 8 | WARMST | 1 | Warmstart-Marke (0 nach dem Einschalten/255 nach System-Reset). |
| 000A | 10 | DOSVEC | 2 | Startzeiger für Disketten-Programme. |

INTERRUPTS

| | | | | |
|---|---|---|---|---|
| 0010 | 16 | POKMSK | 1 | POKEY Interupt Maske. |
| 0042 | 66 | CRITIC | 1 | steht hier eine 1 ist die Tasterwiederholfunktion ausgeschaltet. |

Echtzeituhr (Real-Time-Clock)

| | | | | |
|---|---|---|---|---|
| 0012 | 18 | RTCLOCK | 3 | Interne Uhr. (+0 = MSB; +1 = NSB; +2 = LSB) |

System Uhren(System VBLANK Timer)

| | | | | |
|---|---|---|---|---|
| 0218 | 536 | CDTMV1 | 2 | Wert des System-timer 1. |
| 021A | 538 | CDTMV2 | 2 | Wert des System-timer 2. |
| 021C | 540 | CDTMV3 | 2 | Wert des System-timer 3. |
| 021E | 542 | CDTMV4 | 2 | Wert des System-timer 4. |
| 0220 | 544 | CDTMC5 | 2 | Wert des System-timer 5. |
| 0226 | 550 | CDTMA1 | 2 | Sprungadresse System-timer 1. |
| 0228 | 552 | CDTMA2 | 2 | Sprungadresse System-timer 2. |
| 022A | 554 | CDTMF3 | 2 | Marke System-timer 3. |
| 022C | 556 | CDTMF4 | 1 | Marke System-timer 4. |
| 022E | 558 | CDTMF5 | 2 | Marke System-timer 5. |

NMI-Interrupt-Zeiger (NMI-Interrupt-Vector)

| | | | | |
|---|---|---|---|---|
| 0200 | 512 | VDSLST | 2 | Zeiger zur Display-List-Interrupt-Routine. |
| 0222 | 546 | VVBLKI | 2 | Zeiger zur immediate VBLANK Interrupt-Routine. |
| 0224 | 548 | VVBLKD | 2 | Zeiger zur deferred VBLANK Interrupt-Routine. |

IRQ-Interrupt-Zeiger (IRQ-Interrupt-Vector)

| | | | | |
|---|---|---|---|---|
| 0202 | 514 | VPRCED | 2 | Signal zur Steuerung der seriellen Ein-/Ausgänge. |
| 0204 | 516 | VINTER | 2 | Unterbrechungssignal für serielle Ein-/Ausgänge. |
| 0206 | 518 | VBREAK | 2 | Zeiger zur BREAK-Routine. |
| 0208 | 520 | VKEYBD | 2 | Zeiger für Interrupts der Tastatur. |
| 020A | 522 | VSERIN | 2 | serielle Ein-/Ausgängge bereit zum Datenempfang. |
| 020C | 524 | VSEROR | 2 | serielle Ein-/Ausgängen bereit zum Datensenden. |
| 020E | 526 | VSERC | 2 | Datenübertragung beendet. |



| | | | | |
|---|---|---|---|---|
| 0210 | 528 | VTIMR1 | 2 | Zeiger zum POKEY-timer 1-Interrupt. |
| 0212 | 530 | VTIMR2 | 2 | Zeiger zum POKEY-timer 2-Interrupt. |
| 0214 | 532 | VTIMR4 | 2 | Zeiger zum POKEY-timer 4-Interrupt. |
| 0216 | 534 | VIMIRQ | 2 | genereller IRQ-Zeiger. |

Hardware-Schattenregister (Hardware-Register-Update)

| | | | | |
|---|---|---|---|---|
| 0230 | 560 | SDLSTL | 1 | Adresse der Display-List (Low-Byte). |
| 0231 | 561 | SDLSTH | 1 | Adresse der Display-List (High-Byte). |
| 02C0 | 704 | PCOLRx | 4 | Farb-Register Player. |
| 02C4 | 708 | PCOLORx | 5 | Farb-Register Bildschirmteile. |
| 02F3 | 755 | CHACT | 1 | Zeichendarstellungs-Kontrolle. |
| 02F4 | 756 | CHBAS | 1 | High-Byte des Zeichensatzes im ROM-Bereich. $E0 = Großbuchstaben, $E2 = Kleinbuchstaben, |

Tabelle E.2 Adressen des ROM-Bereichs (Hardware-Adresses)

| ADRESSE HEX | ADRESSE DEZIMAL | NAME | FUNKTION S=schreiben L=lesen | SCHATTEN REGISTER HEX | SCHATTEN REGISTER DEZ | NAME |
|---|---|---|---|---|---|---|
| ANTIC (Antic-Chip) | | | | | | |
| D400 | 54272 | DMACTL | direkter Speicherzugriff (DMA-Kontrolle (S)). | 22F | 559 | SDMCTL |
| D401 | 54273 | CHACTL | Zeichensatzkontrolle (S) | 2F3 | 755 | CHART |
| D402 | 54274 | DLISTL | Display-list-Zeiger (Low-Byte (S)). | 230 | 560 | SDLSTL |
| D403 | 54275 | DLISTH | Display-list-Zeiger (High-Byte (S)). | 231 | 561 | SDLSTH |
| D404 | 54276 | HSCROL | horizontales Verschieben (horizontal scroll (S)). | | | |
| D405 | 54277 | VSCROL | vertikales Verschieben (vertical scroll (S)). | | | |
| D407 | 54279 | PMBASE | Player-Missile-Basis Adresse (S). | | | |
| D409 | 54281 | CHBASE | Zeichen-Basis-Adresse (S). | 2F4 | 756 | CHBAS |
| D40A | 54282 | WSYNC | warten auf horizontale Syncronisation (S). | | | |
| D40B | 54283 | VCOUNT | zählt vertikale Zeilen (L). | | | |
| D40E | 54286 | NMIEN | freigeben des nicht-maskierten Interrupts (S). | | | |
| D40F | 54287 | NMIRES | setzt Interrupt zurück (W). | | | |
| D40F | 54287 | NMIST | Status des Interrupts (L). | | | |

D410-D4FF (54288 - 54527) Wiederholung der Adressen von D400 bis D40F.

ATARINSIDE

CTIA- / GTIA-Chip

KONTROLLE DER PLAYER-MISSILE-GRAFIK

Horizontale Position (schreiben)

| | | | |
|---|---|---|---|
| D000 | 53248 | HPOSP0 | horizontale Position Player0 |
| D001 | 53249 | HPOSP1 | horizontale Position Player1 |
| D002 | 53250 | HPOSP2 | horizontale Position Player2 |
| D003 | 53251 | HPOSP3 | horizontale Position Player3 |
| D004 | 53252 | HPOSM0 | horizontale Position Missile0 |
| D005 | 53253 | HPOSM1 | horizontale Position Missile1 |
| D006 | 53254 | HPOSM2 | horizontale Position Missile2 |
| D007 | 53255 | HPOSM3 | horizontale Position Missile3 |

Kollisionskontrolle (Collision-Control) (lesen)

| | | | |
|---|---|---|---|
| D000 | 53248 | M0PF | Missile 0 / Spielfeld |
| D001 | 53249 | M1PF | Missile 1 / Spielfeld |
| D002 | 53250 | M2PF | Missile 2 / Spielfeld |
| D003 | 53251 | M3PF | Missile 3 / Spielfeld |
| D004 | 53252 | P0PF | Player 0 / Spielfeld |
| D005 | 53253 | P1PF | Player 1 / Spielfeld |
| D006 | 53254 | P2PF | Player 2 / Spielfeld |
| D007 | 53255 | P3PF | Player 3 / Spielfeld |
| D008 | 53256 | M0PL | Missile 0 / Player |
| D009 | 53257 | M1PL | Missile 1 / Player |
| D00A | 53258 | M2PL | Missile 2 / Player |
| D00B | 53259 | M3PL | Missile 3 / Player |
| D00C | 53260 | P0PL | Player 0 / Player |
| D00D | 53261 | P1PL | Player 1 / Player |
| D00E | 53262 | P2PL | Player 2 / Player |
| D00F | 53263 | P3PL | Player 3 / Player |

Kollisionsregister löschen (Collision-Clear)(schreiben)

| | | | |
|---|---|---|---|
| D01E | 53278 | HITCLR | Register löschen |

Größenkontrolle (Size-Controll) (schreiben)
Anmerkung: 0=normale, 1=doppelte, 3=vierfache Größe

| | | | |
|---|---|---|---|
| D008 | 53256 | SIZEP0 | Größe Player 0 |
| D009 | 53257 | SIZEP1 | Größe Player 1 |
| D00A | 53258 | SIZEP2 | Größe Player 2 |
| D00B | 53259 | SIZEP3 | Größe Player 3 |
| D00C | 53260 | SIZEM | Größe für alle Missiles |

Grafik-Register (schreiben)

| | | | |
|---|---|---|---|
| D00D | 53261 | GRAFP0 | Grafik Player 0 |
| D00E | 53262 | GRAFP1 | Grafik Player 1 |
| D00F | 53263 | GRAFP2 | Grafik Player 2 |
| D010 | 53264 | GRAFP3 | Grafik Player 3 |
| D011 | 53265 | GRAFM | Grafik für alle Missiles |



Feuerknöpfe der Steuerknüppel (Joystick-Triggers) (lesen)

| | | | | | | |
|---|---|---|---|---|---|---|
| D010 | 53264 | TRIG0 | Knopf von Joystick (0) | 284 | 644 | STRIG0 |
| D011 | 53265 | TRIG1 | Knopf von Joystick (1) | 285 | 645 | STRIG1 |
| D012 | 53266 | TRIG2 | Knopf von Joystick (2) | 286 | 646 | STRIG2 |
| D013 | 53267 | TRIG3 | Knopf von Joystick (3) | 287 | 647 | STRIG3 |

Kontrolle der Farben (Color-Luminance-Control (schreiben)

| | | | | | | |
|---|---|---|---|---|---|---|
| D012 | 53266 | COLPM0 | Farbe Player/Missile 0 | 2C0 | 704 | PCOLR0 |
| D013 | 53267 | COLPM1 | Farbe Player/Missile 1 | 2C1 | 705 | PCOLR1 |
| D014 | 53268 | COLPM2 | Farbe Player/Missile 2 | 2C2 | 706 | PCOLR2 |
| D015 | 53269 | COLPM3 | Farbe Player/Missile 3 | 2C3 | 707 | PCOLR3 |
| D016 | 53270 | COLPF0 | Farbe vom Spielfeld 0 | 2C4 | 708 | COLOR0 |
| D017 | 53271 | COLPF1 | Farbe vom Spielfeld 1 | 2C5 | 709 | COLOR1 |
| D018 | 53272 | COLPF2 | Farbe vom Spielfeld 2 | 2C6 | 710 | COLOR2 |
| D019 | 53273 | COLPF3 | Farbe vom Spielfeld 3 | 2C7 | 711 | COLOR3 |
| D01A | 53274 | COLBK | Farbe vom Hintergrund | 2C8 | 712 | COLOR4 |

Prioritäts-Kontrolle (Priority-Control) (schreiben)

| | | | | | | |
|---|---|---|---|---|---|---|
| D01B | 53275 | PRIOR | Auswahl der Priorität | 26F | 623 | GPRIOR |

Grafik-Kontrolle (schreiben)

| | | | |
|---|---|---|---|
| D01D | 53277 | GRACTL | Kontrolle der Grafik |

VERSCHIEDENE EIN- / AUSGABEFUNKTIONEN (Miscellaneous I/O Funktions)
PAL / NTSC Test

| | | | |
|---|---|---|---|
| D014 | 53268 | PAL | liest PAL / NTSC bits |

Konsolen-Tasten (Console-Switches)

| | | | |
|---|---|---|---|
| D01F | 53279 | CONSOL | Konsolenregister. |

POKEY-Chip
Ton-Generatoren (Audio) (schreiben)

| | | | |
|---|---|---|---|
| D200 | 53760 | AUDF1 | Tongenerator 1 Frequenz |
| D201 | 53761 | AUDC1 | Tongenerator 1 Kontrolle |
| D202 | 53762 | AUDF2 | Tongenerator 2 Frequenz |
| D203 | 53763 | AUDC2 | Tongenerator 2 Kontrolle |
| D204 | 53764 | AUDF3 | Tongenerator 3 Frequenz |
| D205 | 53765 | AUDC3 | Tongenerator 3 Kontrolle |
| D206 | 53766 | AUDF4 | Tongenerator 4 Frequenz |
| D207 | 53767 | AUDC4 | Tongenerator 4 Kontrolle |

| | | | |
|---|---|---|---|
| D208 | 53768 | AUDCTL | Audiokontrolle |

ATARINSIDE

Start-Timer (schreiben)

| | | | | | | |
|---|---|---|---|---|---|---|
| D209 | 53769 | STIMER | setzt die Frequenz der 4 Tongeneratoren zurück. | | | |

Potentiometer der Drehregler (Pot-Scan-Control)

| | | | | | | |
|---|---|---|---|---|---|---|
| D200 | 53760 | POT0 | liest Poti 0 | 270 | 624 | PADDL0 |
| D201 | 53761 | POT1 | liest Poti 1 | 271 | 625 | PADDL1 |
| D202 | 53762 | POT2 | liest Poti 2 | 272 | 626 | PADDL2 |
| D203 | 53763 | POT3 | liest Poti 3 | 273 | 627 | PADDL3 |
| D204 | 53764 | POT4 | liest Poti 4 | 274 | 628 | PADDL4 |
| D205 | 53765 | POT5 | liest Poti 5 | 275 | 629 | PADDL5 |
| D206 | 53766 | POT6 | liest Poti 6 | 276 | 630 | PADDL6 |
| D207 | 53767 | POT7 | liest Poti 7 | 277 | 631 | PADDL7 |
| D208 | 53768 | ALLPOT | liest Status der 8 Poti's | | | |
| D20B | 53771 | POTGO | startet die Drehregler Sequenz in Verbindung mit dem VBLANK-Zeiger. | | | |

Tastaturkontrolle (Keyboard-Scan and -Control) (lesen)

| | | | | | | |
|---|---|---|---|---|---|---|
| D209 | 53769 | KBCODE | Tastatur-Code | 2FC | 764 | CH |

Zufallszahlengenerator (Random-Number-Generator) (lesen)

| | | | |
|---|---|---|---|
| D20A | 53770 | RANDOM | Zufallszahlengenerator |

Serieller Ein- / Ausgang

| | | | | | | |
|---|---|---|---|---|---|---|
| D20A | 53770 | SKRES | setzt SKSTAT ($D20F) zurück (S). | | | |
| D20D | 53773 | SERIN | serieller Ein-/Ausgang (L / S). | | | |
| D20F | 53775 | SKCTLS | Steuerung der seriellen Ein- / Ausgänge (Schnittstelle). | 232 | 562 | SSKCTL |

IRQ Interrupt

| | | | | | | |
|---|---|---|---|---|---|---|
| D20E | 53774 | IRQEN | IRQ-Interrupt ermöglichen. | 10 | 16 | POKMSK |
| D20F | 53775 | IRQST | IRQ-Interrupt Status (L). | | | |

D210-D2FF (53776 - 54015) haben die gleiche Funktion wie die Adressen von D200-D20F



PIA-Chip

Steuerknüppel-Register (Joystick-Registers)

| | | | | | | |
|---|---|---|---|---|---|---|
| D300 | 54016 | PORTA | schreibt oder liest Daten von den Eingangsbuchsen 1 und 2. | 278 | 632 | STICK0 |
| | | | | 279 | 633 | STICK1 |
| D301 | 54017 | PORTB | schreibt oder liest Daten von den Eingangsbuchsen 3 und 4. | 27A | 634 | STICK2 |
| | | | | 27B | 635 | STICK3 |
| D302 | 54018 | PACTL | Kontrolle des Ausgangs A Steht hier 60, wird der Motor des Programmrekorders ausgeschaltet. Zum Einschalten 52 POKEn. (S). | | | |
| D303 | 54019 | PBCTL | Kontolle des Ausgangs B | | | |

Anmerkung: bei den Ausgängen A und B handelt es sich nicht um Buchsen außerhalb des Computers, sondern um so bezeichnete Ausgänge des PIA-Chips.

D304-D3FF (54020 - 54271) haben die gleiche Bedeutung wie die Adressen von D300-D303 (54016-54019).

ATARINSIDE

# Anhang F PROGRAMM-KONVERTIERUNG

COPYRIGHT-HINWEIS

Computerprogramme sind in der Regel durch gesetzliche Vorschriften kopiergeschützt. Der Käufer eines Programms hat jedoch die Möglichkeit, dieses aus Sicherheitsgründen für sich zu kopieren (Backupcopy). Desweiteren kann er versuchen, das erworbene Programm zu modifizieren, um eine lauffähige Version für das von ihm verwendete Computersystem zu erstellen. Allerdings darf er dieses Programm dann nicht an dritte Personen weitergeben, ohne die Genehmigung des Inhabers der Kopierrechte einzuholen bzw. zu besitzen.

## KONVERTIEREN VON PROGRAMMEN IN ATARI MICROSOFT BASIC II

Das COMMODORE PET' BASIC, APPLE'' APPLESOFT'' BASIC und das RADIO SHACK''' LEVEL II BASIC wurden entwickelt von Microsoft. Die starke Ähnlichkeit dieser BASIC-Dialekte erlaubt es, die Programme von Computer zu Computer auszutauschen. Dazu sind nur einige Änderungen notwendig, die nachfolgend aufgeführt wurden.

Microsoft BASIC wird in vier Abstufungen aufgeteilt: 4k BASIC, 8k BASIC, erweitertes (extended) BASIC und full-BASIC. Diese Abstufungen ergeben sich aus der Anzahl der Befehle und dem zur Verfügung stehenden Speicherbereich. Eine Ursache der Inkompatibilität der Computerprogramme sind die unterschiedlichen Ausbaustufen der verwendeten BASIC-Dialekte. PET BASIC und APPLE APPLESOFT BASIC sind 8k-Typen, während RADIO SHACK LEVEL II und ATARI Microsoft II full-BASIC-Typen sind. Diese Tatsache erleichtert allerdings die Umwandlung in Microsoft BASIC II. Die beiden BASIC-Dialekte haben folgende Unterschiede:

- Daten Typ: im 8k BASIC ist die doppelte Genauigkeit von Zahlen nicht enthalten. Es werden nur 9 Stellen ausgegeben. Ganzzahlige Variablen werden in eine Zahl einfacher Genauigkeit umgewandelt, bevor sie weiterverarbeitet werden. Dadurch sind Berechnungen langsamer.
- Die PRINT USING-Anweisung gibt es beim 8k BASIC nicht. Dadurch müssen Sie sich Ihre Ziffern selbst formatieren.
- IF...THEN...ELSE, DEFINT, DEFSNG, DEFDBL, DEFSTR, TRON, TROFF, RESUME und LINE INPUT sind beim 8k BASIC nicht vorhanden.
- Die Funktionen INSTR und STRING$ sind ebenfalls nicht anwendbar.
- Beim 8k BASIC ist nur die einfache Dimensionierung möglich.
- Vom Anwender definierte Funktionen dürfen beim 8k BASIC nur ein Argument haben.

Die am schwierigsten umzuwandelnden Bereiche sind die vom jeweiligen Computertyp abhängenden Merkmale der Grafikerstellung und dem Gebrauch der Maschinenspracheunterroutinen. In allen Programmen sollten deshalb diese Teile gesondert umgewandelt werden.

'PET ist ein in den USA eingetragenes Markenzeichen der Firma Commodore Business Machines, Inc.
''APPLE und APPLESOFT sind in den USA eingetragene Markenzeichen von APPLE COMPUTER
'''RADIO SHACK ist ein in den USA eingetragenes Markenzeichen der TANDY CORPORATION.



# KONVERTIERUNG VON COMMODORE (PET) BASIC VERSION 4.0 Anhang G

Die meisten Schwierigkeiten bei der Umwandlung von Programmen in Commodore-(PET)-BASIC rühren daher, daß es sich dabei um ein 8k BASIC handelt. In der folgenden Aufstellung finden Sie die bei der Umwandlung zu berücksichtigenden Punkte:

- Der Zeichsatz der Commodore-(PET)-Computer läßt sich um 256 Grafikzeichen erweitern. Dabei handelt es sich ausschließlich um Blockgrafikzeichen. Um diese Zeichen mit Ihrem ATARI-Computer zu simulieren, wandeln Sie den Zeichensatz um.

- Die Variablen TI$ und TI des Commodore-BASIC entsprechen den Variablen TIME$ und TIME. ST steht für die Anweisung STATUS und der griechische Buchstabe PI für die Konstante PI.

- Die Ein- und Ausgänge der Commodore-Computer arbeiten mit speziellen Anweisungen (IEEE bus control). Die Parameter des OPEN-Befehles müssen komplett geändert werden. Die übertragung von Daten findet wie beim ATARI Microsoft BASIC II durch die Anweisungen PRINT und INPUT auf einem spezifizierten Datenkanal statt.

- Der Bildschirm des Commodore PET hat 40 Zeichen in 25 Zeilen. Wenn also Menues dargestellt werden sollen, muß dieser Programmteil u. U. geändert werden.

- PEEK und POKE sind computerabhängige Anweisungen, da hier Eingriffe in den Speicherbereich vorgenommen werden. Beim Commodore wird z.B. die Positionierung des Cursors mit POKE vorgenommen. Jedes PEEK und POKE muß gesondert analysiert und entsprechend neu programmiert werden.

- Commodore-BASIC gibt Befehle zur Cursorsteuerung, (z.B. Bildschirm löschen), mit Text Strings wie das ATARI Microsoft BASIC II. Aber die Funktion der Befehle weichen voneinander ab und muß deshalb geändert werden.

- Die Anweisung CLEAR lautet CLS.

- Werden in dem zu konvertierenden Programm Maschinensprach-Unterroutinen verwendet, müssen diese komplett umgeschrieben werden. Es wird zwar bei beiden Computern der gleiche Prozessor verwendet, aber die Adressen des Speicherbereichs sind verschieden. Der Aufruf der Unterroutine geschieht beim PET mit der Anweisung EXEC.

- Der Commodore-PET-Computer verfügt weder über Tongeneratoren noch über spezielle Grafikmöglichkeiten, so daß in diesem Bereich keine Änderungen erforderlich werden.

- Die Funktion RND ist unterschiedlich. RND mit positivem Vorzeichen gibt eine Zahl zwischen 0 und 1 aus.

Wenn ein eigener Zeichensatz entworfen wurde, der dem des Commodore-PET entspricht, dürfte das Umwandeln von Programmen keine Schwierigkeiten mehr bereiten. Ausnahme sind das Umwandeln von PEEKs und POKEs und evtl. vorhandener Maschinensprache-Unterroutinen.

ATARI INSIDE

## KONVERTIEREN IN ATARI MICROSOFT BASIC II

Benutzen Sie die folgende Tabelle zum Konvertieren von Programmen, die in Commodore-PET-BASIC 4.0 entwickelt wurden.

Anmerkung zur Vereinfachung: die universellen BASIC-Befehle wie RUN und CONT sind gleich und brauchen demnach auch nicht umgewandelt zu werden.

| COMMODORE BEFEHL | ATARI DOS OPTION | ATARI MICROSOFT BASIC II |
|---|---|---|
| DIRECTORY | A<br>DIRECTORY-SEARCH SPEC,LIST FILE?<br>RETURN | |
| COPY | C<br>COPY-FROM,TO?<br>D1:filename,D2:filename | |
| RENAME | E<br>RENAME,GIVE OLD NAME,NEW<br>D2:old filename,new filename | Name |
| SCRATCH | D<br>DELET FILESPEC<br>D1:filename<br>TYPE "Y" TO DELETE filename<br>Y | KILL |
| HEADER | I<br>WHICH DRIVE TO FORMAT? | |



Anhang H

# KONVERTIERUNG TRS-80 RADIO SHACK PROGRAMME IN ATARI MICROSOFT BASIC II

Das Radio Shack BASIC ist ein full-BASIC. Dadurch werden bei einer Konvertierung die Merkmale des Microsoft BASIC II besser genutzt, als bei Apple- oder Commodore-Programmen. Es hat die gleichen Befehle wie das ATARI Microsoft BASIC II, außer dem Befehl COMMON. Das liegt daran, daß Microsoft BASIC II später geschrieben wurde und der Speicherbereich des BASIC's im ATARI-Computer größer ist. Mit Radio Shack BASIC ist das in den Computern Modell I und Modell III installierte BASIC gemeint. Es wird als LEVEL II BASIC bezeichnet. Das BASIC des Modell II ist ähnlich, wurde hier jedoch nicht gesondert aufgeführt.

. Das Bildschirmformat des Radio Shack stellt eines der größten Probleme bei der Konvertierung dar. Es werden 64 Zeichen in 16 Zeilen abgebildet. Programme die alle 64 Zeichen/Zeile benutzen, um z.B. Tabellen oder Menüs darzustellen, müssen in diesen Bereichen komplett umgeschrieben werden.

. Radio Shack benutzt eine Form der Grafik, die auf dem eingebauten schwarz/weiß Bildschirm 128 mal 48 Einzelpunkte darstellt. Diese Grafik kann beliebig mit Zeichen der Tastatur vermischt werden. Es gibt nur vier Anweisungen zum Arbeiten mit dieser Grafik: CLS (löscht Bildschirm), SET (setzt einen Punkt), RESET (löscht einen Punkt) und POINT (Wert eines Bildpunktes).

. Die Ansteuerung eines Druckers erfolgt mit den Anweisungen LPRINT und LLIST. Dabei braucht kein Datenkanal geöffnet zu werden. Die Abspeicherung und das Laden der Programme auf/von Kassette geschieht mit den Anweisungen PRINT und INPUT und der Angabe eines Kanals (1 oder 2). Es können zwei Diskettenlaufwerke angeschlossen werden. Die Datenformate von Kassette und Diskette sind völlig anders als das vom ATARI-Computer verwendete Format.

. Maschinensprach-Unterroutinen werden mit USR aufgerufen. Denken Sie aber daran, daß der Radio Shack Computer mit dem Microprozessor Z-80 arbeitet. Daher müssen diese Programmteile neu geschrieben werden.

. PEEKs und POKEs können nicht direkt übernommen werden.

. Der Cursor wird beim Radio Shack BASIC mit einem SHIFT-8-Zeichen nach dem PRINT positioniert (entsprechend dem AT beim ATARI Microsoft BASIC II).

. Der Fehlercode bei Benutzung der Funktion ERR hat durchgehend eine andere Bedeutung.

ATARINSIDE

| TRS-80 | ATARI | BEDEUTUNG |
|---|---|---|
| CDBL(nr) | - | doppelte Genauigkeit der Ziffer nr. |
| CINT(nr) | - | bildet Ganzzahl, die kleiner ist als als nr (z.B.: 7.52 = 7 / -6.5 = -7). |
| CLOAD | CLOAD<br>LOAD "C:" | lädt BASIC Programme von Kassette. |
| CLOAD? | VERIFY "C:" | vergleicht Programm auf Kassette mit Programm im Speicher. |
| CSNG(X) | automatisch vorhanden | einfache Genauigkeit des Ausdrucks x. |
| EDITzn | LIST zeilennummer | erlaubt das editieren bestimmter, vorgewählter Zeilen. |
| FIX(x) | SGN(x)*INT(ABS(x)) | unterdrückt die Stellen hinter dem Komma. |
| INPUT#-1 | OPEN #5,"C:" INPUT<br>INPUT #5 | liest Daten von einer Kassette. |
| LLIST | LIST"P:"nr-no | listet Programm auf Drucker aus. |
| LPRINT | OPEN #4,"P:" OUTPUT<br>PRINT #4,"Text" | druckt Text aus. |
| PRINT MEM | PRINT FRE(0) | gibt verbleibenden Speicherplatz an. |
| POINT(X,Y)<br>POINT(X,Y) | PUT #x,AT(s,b)<br>GET #x,AT(s,b) | schreibt Daten an bestimte Position.<br>liest Daten von bestimten Sektor/Byte. |
| PRINT n,list | PRINT #6,AT(X,Y);list | |
| PRINT#-1 | OPEN*3,"C:"OUTPUT<br>PRINT*3 | schreibt Daten auf Kassette. |
| RANDOM | RANDOMIZE | bestimmt RND Funktion. |



Anhang I

# KONVERTIERUNG APPLESOFT PROGRAMME IN ATARI MICROSOFT BASIC II

Apple verwendet in seinen Computern den gleichen BASIC Typ (8k Ausführung) wie Commodore (PET). Deshalb sind auch die gleichen Punkte beim Konvertieren dieser Programme zu beachten (siehe Anlage G). Es bestehen noch folgende Unterschiede zum Microsoft BASIC II:

. Apple hat die verwendete Sprache Applesoft um zwei Anweisungen erweitert. Es handelt sich dabei um ONERR und POP. Bei ONERR handelt es sich um eine Anweisung zum Festlegen der Programmverzweigung in einem Fehlerfall. Sie kann einfach gegen die im ATARI Microsoft BASIC II enthaltene Anweisung ON ERROR ausgetauscht werden. Für die Anweisung POP besteht kein alternatives Programmierwort. Es ist für den Fall einer unstrukturierten Programmierung gedacht, bei der ein Unterprogramm kein wirkliches Unterprogramm ist, weil es mit einem GOTO verlassen wird. Bei einem GOSUB wird ein Zeiger im Stapel gesetzt, der auf die Zeile hinweist, in der die GOSUB-Anweisung steht. Dies ist erforderlich, damit das RETURN an dieser Stelle die Programmausführung fortsetzt. Wird das Unterprogramm mit einem GOTO verzweigt, muß dieser Zeiger zurückgesetzt werden, damit das nächste RETURN richtig ausgeführt wird. Dies geschieht mit dem POP-Befehl. Sie können ihn durch Setzen eines Flags simulieren, wenn es erforderlich sein sollte.

. Der im Apple verwendete Ausgabebildschirm unterscheidet sich von dem des ATARI-Computer. Werden also Tabellen oder Menüs auf dem genannten Display dargestellt, müssen diese Programmteile geändert werden.

. Die Kontrolle der Ein- und Ausgänge erfolgt beim Apple-Computer über die Aktivierung der eingestecken Pheripheriekarten. Daher müssen Ein- / Ausgaberoutinen entprechend geändert werden (OPEN,CLOSE).

. Die am schwierigsten zu konvertierenden Programmpassagen sind die Bereiche der SOUND- und GRAFIK-Befehle. Die höchste Grafikauflösung beträgt beim Apple Computer 280 mal 192 Punkte. Dabei ist es nicht möglich, einem Punkt unabhängig von den anderen eine bestimmte Farbe zuzuweisen (Apple verwendet 8 Farben). Der Tonausgang besteht aus einem Tongenerator, der den im Computer eingebauten Lautsprecher anspricht.

ATARINSIDE

# Anhang J KONVERTIERUNG ATARI 8K BASIC IN ATARI MICROSOFT BASIC II

Das ATARI Microsoft BASIC II hat verbesserte Anweisungen zur Kontrolle und Erstellung von Grafiken. Die vorhandenen Grafikbefehle, insbesondere bei der Verwendung von Player/Missile-Objekten muß umgeschrieben werden. So sind die SETCOLOR-Register 0,1,2 und 3 jetzt für die Farbgebung der PM-Objekte verantwortlich. Die SETCOLOR-Anweisungen 0, col, lum entsprechen nun den SETCOLOR-Anweisungen 4, col, lum. Das bedeutet, daß die Farbregister 0,1,2,3 und 4 jetzt mit 4,5,6,7 und 8 bezeichnet sind. Weiterhin wird die FILL-Anweisung umgeschrieben und die DRAWTO-Anweisung gegen PLOT...TO ausgetauscht.

Microsoft verändert auch das STRING-Handling. Wenn die Initialisierung eines Programm zu viel Speicherplatz benötigt, sollten Sie die Konvertierung in Microsoft BASIC II erwägen.

In der nachfolgenden Tabelle, sind noch einmal die unterschiedlichen Programmierworte gegenübergestellt. Programme, die im LIST-Format abgespeichert wurden, können eingeladen und dann geändert werden. So ist auch ein Testen der jeweiligen Programmänderungen möglich.

| ATARI 8k BASIC | ATARI Microsoft BASIC II | Kommentar |
|---|---|---|
| ADR(VAR$) | VARPTR(VAR$)+1 | Identische Anweisung. |
| CLR | CLEAR | |
| DEG | -- | keine gleichwertige Funktion. |
| PLOT X,Y<br>DRANTO X,Y | PLOT X,Y TO X,Y | zeichnet Linie auf Bildschirm im Grafikmodus. |
| LIST nr,no | LIST nr-no | |
| LOCATE X,Y,VAR | VAR=ASC(SCRN$(X,Y)) | liest Wert in angegebener X,Y Position. |
| LPRINT "Text" | OPEN #7,"P:" OUTPUT<br>PRINT #7,"Text" | druckt Text aus. |
| OPEN #nr,<br>nr1,nr2,<br>Datenkanal | OPEN #nr,<br>Datenkanal INPUT | |
| POINT #nr,<br>Sektor,Byte | INPUT #nr,<br>AT(Sektor,Byte)<br>oder<br>PRINT #nr,<br>AT(Sektor,Byte) | |



| ATARI 8k BASIC | ATARI Microsoft BASIC II | Kommentar |
|---|---|---|
| POP | -- | benutzen Sie eine Maschinensprache Unterroutine, die mit USR aufgerufen wird und den Stapelzeiger zurücksetzt. |
| POSITION X,Y<br>PRINT #6; | PRINT #6,AT(X,Y) | |
| SOUND Stimme,<br>Ton,Verzerrung<br>Lautstärke | SOUND Stimme,<br>Ton,Verzerrung,<br>Lautstärke,Dauer | die Angabe der Dauer eines Tons ist eine Erweiterung. Sie wird in 1/50 Sekunden angegeben. Das Zurücksetzen der Tongeneratoren mit der Anweisung SOUND Stimme, 0,0,0 kann dadurch entfallen. |
| TRAP Verzweigung | ON ERROR Verzweigung | |
| USR(Adresse,Liste) | USR(Adresse,<br>Pointer) | Sie können nach der USR-Funktion nur noch ein Argument angeben. |
| XIO 18 | FILL X1,Y1 TO<br>X2,Y2 | Microsoft FILL zeichnet die Linie X1,Y1 zur Position X2,Y2. |

PADDLE, PTRIG, STICK und STRIG werden beim Microsoft BASIC II über PEEK-Anweisungen ausgewertet (siehe auch Abschnitt 6).

ATARINSIDE

## Anhang K ATASCII ZEICHENSATZ

| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | |
| 2 | 2 | |
| 3 | 3 | |
| 4 | 4 | |
| 5 | 5 | |
| 6 | 6 | |
| 7 | 7 | |
| 8 | 8 | |
| 9 | 9 | |
| 10 | A | |
| 11 | B | |
| 12 | C | |
| 13 | D | |
| 14 | E | |
| 15 | F | |
| 16 | 10 | |
| 17 | 11 | |
| 18 | 12 | |
| 19 | 13 | |
| 20 | 14 | |
| 21 | 15 | |
| 22 | 16 | |
| 23 | 17 | |
| 24 | 18 | |
| 25 | 19 | |
| 26 | 1A | |



| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 27 | 1B | |
| 28 | 1C | ↑ |
| 29 | 1D | ↓ |
| 30 | 1E | ← |
| 31 | 1F | → |
| 32 | 20 | |
| 33 | 21 | ! |
| 34 | 22 | " |
| 35 | 23 | # |
| 36 | 24 | $ |
| 37 | 25 | % |
| 38 | 26 | & |
| 39 | 27 | ' |
| 40 | 28 | ( |
| 41 | 29 | ) |
| 42 | 2A | * |
| 43 | 2B | + |
| 44 | 2C | , |
| 45 | 2D | - |
| 46 | 2E | . |
| 47 | 2F | / |
| 48 | 30 | 0 |
| 49 | 31 | 1 |
| 50 | 32 | 2 |
| 51 | 33 | 3 |
| 52 | 34 | 4 |
| 53 | 35 | 5 |
| 54 | 36 | 6 |

ATARINSIDE

| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 55 | 37 | 7 |
| 56 | 38 | 8 |
| 57 | 39 | 9 |
| 58 | 3A | : |
| 59 | 3B | ; |
| 60 | 3C | < |
| 61 | 3D | = |
| 62 | 3E | > |
| 63 | 3F | ? |
| 64 | 40 | @ |
| 65 | 41 | A |
| 66 | 42 | B |
| 67 | 43 | C |
| 68 | 44 | D |
| 69 | 45 | E |
| 70 | 46 | F |
| 71 | 72 | G |
| 72 | 48 | H |
| 73 | 49 | I |
| 74 | 4A | J |
| 75 | 4B | K |
| 76 | 77 | L |
| 77 | 4D | M |
| 78 | 4E | N |
| 79 | 4F | O |
| 80 | 50 | P |
| 81 | 51 | Q |
| 82 | 52 | R |
| 83 | 53 | S |



| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 84 | 54 | T |
| 85 | 55 | U |
| 86 | 56 | V |
| 87 | 57 | W |
| 88 | 58 | X |
| 89 | 59 | Y |
| 90 | 5A | Z |
| 91 | 5B | [ |
| 92 | 5C | \ |
| 93 | 5D | ] |
| 94 | 5E | ^ |
| 95 | 5F | _ |
| 96 | 60 | ♦ |
| 97 | 61 | a |
| 98 | 62 | b |
| 99 | 63 | c |
| 100 | 64 | d |
| 101 | 65 | e |
| 102 | 66 | f |
| 103 | 67 | g |
| 104 | 68 | h |
| 105 | 69 | i |
| 106 | 6A | j |
| 107 | 6B | k |
| 108 | 6C | l |
| 109 | 6D | m |
| 110 | 6E | n |
| 111 | 6F | o |
| 112 | 70 | p |

ATARINSIDE

| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 113 | 71 | q |
| 114 | 72 | r |
| 115 | 73 | s |
| 116 | 74 | t |
| 117 | 75 | u |
| 118 | 76 | v |
| 119 | 77 | w |
| 120 | 78 | x |
| 121 | 79 | y |
| 122 | 7A | z |
| 123 | 7B | ♠ |
| 124 | 7C | \| |
| 125 | 7D | ↖ |
| 126 | 7E | ◀ |
| 127 | 7F | ▶ |
| 128 | 80 | ♥ |
| 129 | 81 | ├ |
| 130 | 82 | │ |
| 131 | 83 | ┘ |
| 132 | 84 | ┤ |
| 133 | 85 | ┐ |
| 134 | 86 | ╱ |
| 135 | 87 | ╲ |
| 136 | 88 | ◢ |
| 137 | 89 | ▗ |
| 138 | 8A | ◣ |
| 139 | 8B | ▝ |
| 140 | 8C | ▘ |
| 141 | 8D | ▔ |



| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 142 | 8E | ▁ |
| 143 | 8F | ▖ |
| 144 | 90 | ♣ |
| 145 | 91 | ┌ |
| 146 | 92 | ─ |
| 147 | 93 | ┼ |
| 148 | 94 | ● |
| 149 | 95 | ▬ |
| 150 | 96 | ▌ |
| 151 | 97 | ┬ |
| 152 | 98 | ┴ |
| 153 | 99 | ▐ |
| 154 | 9A | └ |
| 155 | 9B | ⏎ |
| 156 | 9C | ↑ |
| 157 | 9D | ↓ |
| 158 | 9E | ← |
| 159 | 9F | → |
| 160 | A0 | |
| 161 | A1 | ! |
| 162 | A2 | " |
| 163 | A3 | # |
| 164 | A4 | $ |
| 165 | A5 | % |
| 166 | A6 | & |
| 167 | A7 | ' |
| 168 | A8 | ( |
| 169 | A9 | ) |
| 170 | AA | * |

ATARINSIDE

| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 171 | AB | + |
| 172 | AC | , |
| 173 | AD | — |
| 174 | AE | . |
| 175 | AF | / |
| 176 | B0 | 0 |
| 177 | B1 | 1 |
| 178 | B2 | 2 |
| 179 | B3 | 3 |
| 180 | B4 | 4 |
| 181 | B5 | 5 |
| 182 | B6 | 6 |
| 183 | B7 | 7 |
| 184 | B8 | 8 |
| 185 | B9 | 9 |
| 186 | BA | : |
| 187 | BB | ; |
| 188 | BC | < |
| 189 | BD | = |
| 190 | BE | > |
| 191 | BF | ? |
| 192 | C0 | @ |
| 193 | C1 | A |
| 194 | C2 | B |
| 195 | C3 | C |
| 196 | C4 | D |
| 197 | C5 | E |
| 198 | C6 | F |
| 199 | C7 | G |



| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 200 | C8 | H |
| 201 | C9 | I |
| 202 | CA | J |
| 203 | CB | K |
| 204 | CC | L |
| 205 | CD | M |
| 206 | CE | N |
| 207 | CF | O |
| 208 | D0 | P |
| 209 | D1 | Q |
| 210 | D2 | R |
| 211 | D3 | S |
| 212 | D4 | T |
| 213 | D5 | U |
| 214 | D6 | V |
| 215 | D7 | W |
| 216 | D8 | X |
| 217 | D9 | Y |
| 218 | DA | Z |
| 219 | DB | [ |
| 220 | DC | \ |
| 221 | DD | ] |
| 222 | DE | ^ |
| 223 | DE | — |
| 224 | E0 | ♦ |
| 225 | E1 | a |
| 226 | E2 | b |
| 227 | E3 | c |
| 228 | E4 | d |

ATARINSIDE

| Dezimal Code | Hexadezimal Code | Code Zeichen |
|---|---|---|
| 229 | E5 | e |
| 230 | E6 | f |
| 231 | E7 | g |
| 232 | E8 | h |
| 233 | E9 | i |
| 234 | EA | j |
| 235 | EB | k |
| 236 | EC | l |
| 237 | ED | m |
| 238 | EE | n |
| 239 | EF | o |
| 240 | F0 | p |
| 241 | F1 | q |
| 242 | F2 | r |
| 243 | F3 | s |
| 244 | F4 | t |
| 245 | F5 | u |
| 246 | F6 | v |
| 247 | F7 | w |
| 248 | F8 | x |
| 249 | F9 | y |
| 250 | FA | z |
| 251 | FB | ♠ |
| 252 | FC | \| |
| 253 | FD | ↳ |
| 254 | FE | ◀ |
| 255 | FF | ▶ |



RAUM FÜR IHRE EINTRAGUNGEN

ATARINSIDE

# Anhang L VERWENDUNG DER CIOUSR-ROUTINE

Auf der Microsoft BASIC II-Erweiterungsdiskette befinden sich drei USR-Routinen. Diese erlauben eine flexibe Handhabung und Steuerung des Eingangs- / Ausgangsbereichs im ATARI Privat Computer System. Es besteht damit die Möglichkeit Daten direkt an die Eingabe / Ausgabe-Kontrollblöcke (IOCB) zu senden, bzw. von diesen zu empfangen. Die Funktion dieser Kontrollblöcke wird detailiert in den ATARI Home Computer System Technical Reference Notes beschrieben.

Die Routinen geben bei Bedarf das Disketten-Inhaltsverzeichnis aus, formatieren eine Diskette oder rufen einen speziellen IOCB auf. Im folgenden ist angegeben, wie Sie diese Routinen in Ihre Programme einarbeiten können.

## Schritt 1. Einlesen der Routinen in ein BASIC-PROGRAMM

Alle drei Routinen sind auf der Diskette in einem Datenfile zusammengefaßt. Dieses wird mit CIOUSR aufgerufen. Da es sich hierbei um Maschinensprachedaten handelt, werden diese in den Speicherbereich gePOKEt. Zur Reservierung des benötigten Speicherplatzes benutzen Sie die Anweisung OTION RESERVE 160. Die Startadresse dieses reservierten Bereichs ermitteln Sie unter Verwendung von ADDR=VARPTR(RESERVE). Dann geben Sie folgende Zeilen ein.:

```
10 OPEN #1,"D:CIOUSR" INPUT
20 FOR T=0 TO 159
30 GET #1,A
40 POKE ADDR+T,A:NEXT
50 CLOSE #1
```

## Schritt 2. Bestimmen der Datenfelder

Die Routine befindet sich nun im reservierten Bereich des BASIC-Programms. Die drei Routinen heißen PUTIOCB, CALLIOCB und GETIOCB. Dabei ist die Startposition von PUTIOCB ADDR, die von CALLIOCB ADDR+61, GETIOCB startet bei ADDR+81.

Mit GETIOCB werden Daten von einem spezifizierten Eingabe- / Ausgabe-Kontrollblock empfangen und in ein Variablenfeld mit der Länge von 10 übergeben. Sie können einige dieser Werte verändern und diese wieder an den IOCB senden. Dazu benutzen Sie die PUTIOCB-Routine. Wenn Sie Ihre eigenen Parameter auf diese Art gesetzt haben, werden durch CALLCIO diese Werte bei der Aufführung von Datenein- und -ausgaben benutzt.

Der nächste Schritt ist die Dimensionierung eines Feldes für ganzzahlige Variablen. Dort werden die Parameter des IOCB festgelegt. Dieses Feld muß mit der Länge 10 dimensioniert und mit OPTION BASE 0 initialisiert werden.

Auf der nächsten Seite sind die Werte und ihre Bedeutung aufgeführt:



| ELEMENT NUMMER | IOCB PARAMETER |
|---|---|
| 0 | enthält die Nummer des benutzten Datenkanals. |
| 1 | Befehlscode. |
| 2 | STATUS. |
| 3 | Puffer Adresse. |
| 4 | Puffer Länge. |
| 5-10 | Hilfbyte 1-6. |

## Schritt 3. Aufrufen der USR-Routinen

Die CIOUSR-Routinen werden mit der USR-Anweisung aufgerufen. Die GETIOCB-Routine enthält nach der Rückkehr ins BASIC-Programm die aktuellen Werte der IOCB-Parameter. Durch Wechseln dieser Parameter im Variablenfeld können Sie eigene CIO-Funktionen erstellen (z.B ist die Erhöhung der Baudrate bei Verwendung der RS-232 Schnittstelle möglich).

Dazu werden die geänderten Werte mit PUTIOCB in den gewählten IOCB übertragen. Wenn nun CALLCIO aufgerufen wird, führt die Ein- / Ausgabe-Zentraleinheit (CIO) diese Funktionen aus.

Mit der folgenden Zeile können Sie jede der Routinen aufrufen:

```
nvar = USR(Addresse,(VARPTR(array(0)))
```

Dabei bedeuten:

nvar = eine numerische Variable, die nach der Ausführung der Anweisung CALLCIO den Status der CIO-Funktion enthält; nach Ausführung anderer Anweisungen enthält diese Variable keinen Wert.

Adresse = ist die Startadresse der gewünschten Routine (ADDR - PUTIOCB, ADDR+61 - CALLCIO, ADDR+81 - GETIOCB).

array(0) = ist das Feld der ganzzahligen Variablen, die vom Programm benutzt wird, um die Daten für die Routinen abzulegen. Benutzen Sie die Anweisung VARPTR mit dem Element 0 dieses Feldes, um der Routine mitzuteilen, wo Sie mit dem Einlesen dieser Werte beginnen soll (IOCB-Nummer wird als erster Wert eingelesen).

ATARINSIDE

Das folgende Beispielprogramm enthält die Anweisungen zum Benutzen des RS-232-Interfaces für Zwecke der Telekommunikation. Auch das erste Beispielprogramm in Anlage A benutzt die CIOUSR-Routine. Um das folgende Programm problemlos verwenden zu können, laden Sie zunächst den RS-232-Treiber von Diskette (Programmname RS323.SYS). Dazu kopieren Sie das Progamm an das auf der BASIC-Erweiterungsdiskette vorhandene AUTORUN.SYS-File (mit APPEND).

```
C: SELECT ITEM OR RETURN FOR MENU
S: C (RETURN)

C: COPY -- FROM,TO?
S: RS232.SYS,AUTORUN.SYS, A (RETURN)

C: TYPE "Y" IF OK TO USE PROGRAM AREA
   CAUTION: A "Y"
   INVALIDATES MEM.SAV
S: Y (RETURN)
```

```
100 !
110 !ROUTINE ZUM INITILIZIEREN DES RS-232 AUSGANGS
120 !
130 !                                !ADRESSE ERMITTELN
140 DIM CIO%(10),S%(10)
150 OPTION RESERVE 200
160 ADDR=VARPTR(RESERVE)            !SPEICHERT PUFFERADRESSE
170 PUTIOCB=ADDR
180 CALLCIO=ADDR+61
190 PUTIOCB=ADDR+81
200 OPEN #1,"D:CIOUSR" INPUT        !EINLESEN DER ROUTINE IN
                                    DEN RESERVIERTEN BEREICH
210 FOR I=0 TO 159
220 GET #1,D:POKE ADDR+I,D
230 NEXT
240 CLOSE #1
250 !                               !SETZT PARAMETER
260 OPEN #1,"K:" INPUT
270 CIO%(0)=2                       !IOCB NUMMER
280 CIO%(1)=3                       !BEFEHLSCODE OPEN
290 Y=VARPTR(CIO%(3))               !BUFFER ADRESSE
300 FSPEC$="R:"
310 Z=VARPTR(FSPEC$)
315 Y=VARPTR(CIO%(3))
320 POKE Y,PEEK (Z+2)               !STARTADRESSE FUER
                                    DATENPUFFER.
330 POKE Y+1,PEEK (Z+1)
335 Y=VARPTR(S%(3))
340 CIO%(5)=13                      !HILFSBYTE 1 PARAMETER
350 !
360 A=USR(PUTIOCB,VARPTR(CIO%(0)))  !INITIALISIERT IOCB
370 A=USR(CALLCIO,VARPTR(CIO%(0)))
380 A=USR(GETIOCB,VARPTR(CIO%(0)))
390 !
```



```
400 CIO%(1)=40                      !START EIN-/AUSGABE
410 CIO%(5)=13                      !SETZT HILFSBYTE AUF 0
470 A=USR(PUTIOCB,VARPTR(CIO%(0)))  !WECHSELT IOCB
480 A=USR(CALLCIO,VARPTR(CIO%(0)))  !FUEHRT CIO AUS
490 A=USR(PUTIOCB,VARPTR(CIO%(0)))
500 !                               !START DER KOMMUNIKATION
520 PRINT "STARTSCHLEIFE"
530 GET #1,A
540 PUT #2,A
550 POKE 765,255                    !LOESCHT TASTATUR PUFFER
560 X=USR(CALLCIO,VARPTR(S%(0)))    !PRUEFT STATUS RS-232
570 IF PEEK(747)=0 THEN 600         !KEINE ZEICHEN GESENDET
580 GET #2,D
590 IF D<> 10 THEN ? CHR$(D);
600 IF PEEK(764)<>255 THEN 530      !ES WURDEN ZEICHEN EINGEGEBEN
610 GOTO 560
```

ATARINSIDE

Anhang M

# BEI BEENDIGUNG EINES PROGRAMMES

| Taste drücken oder Ausführen der der Anweisung | schließt alle geöffneten Files | Beeinflussung des Stapel - Zeigers | Zurücksetzen angesprochener Tongeneratoren |
|---|---|---|---|
| STOP | nein | nein | ja |
| Fehlerfall | nein | nein | ja |
| BREAK-Taste | nein | nein | ja |
| Beenden des Programms letzte Anweisung oder END | ja | ja | ja |
| Nach einer direkten Eingabe einer Anweisung | nein | ja | nein |
| Anweisung RUN | ja | nein | ja |

ANMERKUNGEN:

1. ATASCII steht für ATARI-ASCII. Buchstaben und Ziffern haben den gleichen Wert wie beim normalen ASCII (American Standard Code for Information Interchange), nur die Werte der Grafikzeichen (CTRL-Zeichen) differieren.

2. Um den Wert von Kleinbuchstaben zu ermitteln, zählen Sie zu dem Wert des Großbuchstaben 32 dazu (A=65,a=97).

3. Zum Ermitteln des ATASCII-Code geben Sie im Direktmodus PRINT ASC("x") ein. x steht für den entsprechenden Buchstaben bzw. die zu ermittelnde Ziffer.

4. Die Zeichen werden normalerweise weiß auf blauem Hintergrund dargestellt. Nach Betätigen der ATARI-(Invers)-Taste werden die Zeichen blau auf weißem Hintergrund dargestellt.



ANHANG N

# ALPHABETISCHE AUFSTELLUNG DER PROGRAMMIERWORTE

| PROGRAMMIER WORT | FUNKTION |
|---|---|
| ABS | Funktion ermittelt den absoluten Wert einer Variablen oder einer errechneten Zahl<br>Beispiel: Y=ABS(A+B) |
| AFTER | Wird gesetzt, um die Ausführung einer Anweisung nach einer bestimmten Zeit zu erreichen. Angegeben wird die Dauer in Anzahl der 1/50 Sekunden. Dieser Wert kann auch als Variable oder als Ergebnis einer Berechnung angegeben werden.<br>Beispiel: AFTER 180 GOTO 1000 |
| AND | Logische Operation. Der Vergleich trifft nur zu, wenn alle Werte im Vergleich zutreffen. UND-Verknüpfung.<br>Beispiel: IF A=30 AND B=30 THEN 1000 |
| ASC | Die in Klammer stehenden Zeichen werden im ATASCII-Code, also als Zahl ausgegeben.<br>Beispiel: PRINT ASC ("A") |
| AT | gibt Daten an Diskette oder Bildschirm an vorbestimmter Position aus.<br>Beispiel: PRINT AT(Sektor,Byte)"Starte hier." |
| ATN | Arcustangens einer Zahl oder eines Ausdrucks in Radiant.<br>Beispiel: PRINT ATN(A) |
| AUTO | Bildet automatisch die Zeilennummern.<br>Beispiel: AUTO 100,50 |
| BASE | wird mit OPTION benutzt und legt den Beginn eines Variablenfeldes fest.<br>Beispiel: OPTION BASE 1 |
| CHR | wird mit OPTION benutzt und reserviert Speicherbereich zur Erstellung eines eigenen Zeichensatzes. Dabei werden mit CHR1 1024 Byte, mit CHR2 512 Byte reserviert. Mit CHR0 wird die Reservierung aufgehoben.<br>Beispiel: OPTION CHR2 |
| CHR$ | mit dieser Funktion wird ein bestimmtes ATASCII-Zeichen ausgegeben. Die Zahl in Klammern kann zwischen 0 und 255 liegen.<br>Beispiel: PRINT CHR$(65) |

ATARINSIDE

| PROGRAMMIER WORT | FUNKTION |
|---|---|
| CLEAR | setze alle Variablen auf Null und löscht alle Strings.<br>Beispiel: CLEAR |
| CLEAR STACK | setzt alle Zeiger des Zeitstapels auf Null.<br>Beispiel: CLEAR STACK |
| CLOAD | lädt Programm von Kassette in den Speicher.<br>Beispiel: CLOAD |
| CLOSE | schließt die geöffneten Datenkanäle.<br>Beispiel: CLOSE #6 |
| CLS | löscht den Textteil des Bildschirms und ändert die Farbe des Hintergrunds, auf den hinter der Anweisung stehenden Wert (Angabe muß nicht gesetzt werden).<br>Beispiel: CLS 180 |
| COLOR | wählt das Farbregister aus, mit dem die FILL- oder PLOT-Anweisung ausgeführt wird.<br>Beispiel: COLOR 2 |
| COMMON | übernimmt die Werte von Variablen, wenn ein neues Programm geladen wird.<br>Beispiel: COMMON A,B,A$ |
| CONT | setzt Programmausführung nach BREAK oder STOP fort.<br>Beispiel: CONT |
| COS | Kosinus einer Variablen oder Ausdrucks in Radiant.<br>Beispiel: A=COS(2.3) |
| CSAVE | speichert das im Speicherbereich befindliche Programm auf die Kassette.<br>Beispiel: CSAVE |
| DATA | bestimmt Daten, die aus einer READ-Anweisung gelesen werden können.<br>Beispiel: 24,34,"plus",6 |
| DEF | Anweisung hat zwei Funktionen:<br>1. Definiert eine arithmetische- oder String-Funktion.<br>Beispiel: DEF WURZEL(X,Y)=SQR(X*X+Y*Y)<br><br>2. Definiert Typ der Variablen als INT, SNG, DBL oder STR.<br>Beispiel: DEFINT A-C |
| DEL | löscht Programmzeilen.<br>Beispiel: DEL 20-25 |



| PROGRAMMIER WORT | FUNKTION |
|---|---|
| DIM | Reserviert einen bestimmen Bereich des Speichers für die Verwendung von Variablenfeldern, -matrizen und Stringfeldern.<br>Beispiel: DIM A(3),B$(10,2,5) |
| END | beendet die Programmausführung, schließt alle Files und kehrt in den Eingabemodus zurück.<br>Beispiel: END |
| EOF | enthält den Wert -1, wenn alle Daten des Files ausgelesen sind.<br>Beispiel: IF EOF(1) GOTO 300 |
| ERL | gibt Zeile an, in der ein Fehler auftrat.<br>Beispiel: PRINT ERL |
| ERR | gibt Codenummer des Fehlers aus.<br>Beispiel: IF ERR=62 THEN END |
| ERROR | simuliert einen Fehler mit der entsprechenden Codenummer, um das Verhalten des Programms im Fehlerfall zu testen.<br>Beispiel: ERROR 170 |
| EXP | errechnet natürliche Exponentialfunktion.<br>Beispiel: B=EXP(A) |
| FILL | zeichnet einen vorbestimmten Bereich einer Grafik aus.<br>Beispiel: FILL 10,10 TO 20,20 |
| FOR..TO..STEP | wiederholt bestimmte Programmsequenzen in Verbindung mit der Anweisung NEXT, bis die Variable den geforderten Wert erreicht hat.<br>Beispiel: FOR I=1 TO 2304 STEP 30 |
| FRE(0) | gibt den dem Programmierer zur Verfügung stehenden Speicherbereich in Byte an.<br>Beispiel: PRINT FRE(0) |
| GET | liest ein Byte von einem geöffneten Datenkanal.<br>Beispiel: GET#1,D |
| GOSUB | verzweigt die Programmausführung zu einem Unterprogramm, das mit der genannten Zeilennummer beginnt.<br>Beispiel: GOSUB 2100 |
| GOTO | verzweigt die Programmausführung zur genannten Zeilennummer.<br>Beispiel: GOTO 100 |
| GRAPHICS | gibt an, welche der Grafikstufen auf dem Bildschirm dargestellt werden soll.<br>Beispiel: GRAPHICS 5+16 |

ATARINSIDE

| PROGRAMMIER WORT | FUNKTION |
|---|---|
| IF...THEN | vergleicht Werte. Trifft der Vergleich zu, wird die hinter THEN stehende Anweisung ausgeführt. Sonst wird das Programm mit der nächsten Zeile fortgesetzt. Beispiel: IF FRE(0) < 1000 THEN PRINT "Speicher voll" |
| IF.THEN.ELSE | wie IF...THEN, nur daß die Programmausführung bei nicht zutreffendem Vergleich mit der hinter ELSE stehenden Zeilennummer fortgesetzt wird. Beispiel: IF X<Y THEN 1000 ELSE GOTO 500 |
| INKEY$ | enthält Zeichen, das über Tastatur eingegeben wird als STRING. Beispiel: A$=INKEY$ |
| INPUT# | liest Daten von einem geöffnetem Datenkanal. Beispiel: INPUT#2,A,B |
| INPUT | weist einer Variablen den über die Tastatur eingegebenen Wert oder einem STRING den eingegebenen Text zu. Beispiel: INPUT "Preis";A:INPUT "Name";A$ |
| INSTR | überprüft, ob ein kleinerer STRING in einem größeren ab dem angegeben Zeichen enthalten ist. Beispiel: INSTR(3,A$,B$) |
| INT | gibt die größte Ganzzahl an, die kleiner als die Variable bzw. das Ergebniss der Berechnung ist. Beispiel: C=INT(X/4.5) |
| KILL | löscht das angegebene Programm von der Diskette. Beispiel: KILL "D:PROGRAM.BAS" |
| LEFT$ | gibt die Zeichen des STRINGs beginnend von links aus. Die Ziffer bestimmt bis zu welchem Zeichen ausgegeben wird. Beispiel: B$=LEFT$(X$,8) |
| LEN | gibt die Anzahl der Zeichen eines STRINGs an. Beispiel: PRINT LEN(B$) |
| LET | weist einer Variablen einen Wert zu. Beispiel: LET A=23.567 |
| LINE INPUT | hat die gleiche Funktion wie INPUT. Nur ist hier die Eingabe von Doppelpunkten, Komma usw. erlaubt. Die Aufforderung zur Eingabe wird nicht mit Fragezeichen gestellt. Beispiel: LINE INPUT "Name ?";N$ |
| LIST | gibt die geforderten Zeilennummern des Programms an Bildschirm / Drucker / Diskettenlaufwerk aus. Beispiel: LIST 100-23000 |



| PROGRAMMIER WORT | FUNKTION |
|---|---|
| LOAD | lädt ein Programm in den Speicherbereich des Computers. Beispiel: LOAD "D:PROGRAM.BAS" |
| LOCK | blockiert ein Programm gegen Überschreiben oder Löschen. Beispiel: LOCK "D:TEST.1" |
| LOG | ermittelt den natürlichen Logarithmus eines Ausdrucks. Beispiel: D=LOG(Y-3) |
| MERGE | vermischt ein Programm mit dem bereits im Speicher stehenden. Beispiel: MERGE "D:SUB1.BAS" |
| MID$ | gibt Zeichen eines Strings von der angegebenen Position aus. Die zweite Zahl gibt an, wieviel Zeichen ausgegeben werden sollen. Beispiel: A$=MID$(X$,5,10) |
| MOVE | bewegt eine bestimmte Anzahl Bytes von einem Bereich des Speichers in einen anderen. Beispiel: MOVE 45000,50000,6 |
| NAME | ändert den Programmnamen auf der Diskette. Beispiel: NAME "D:oldfile" TO "newfile" |
| NEW | löscht Programm im Speicher einschließlich aller Variablen. Beispiel: NEW |
| NEXT | arbeitet zusammen mit der FOR-Anweisung. Beispiel: NEXT I |
| NOT | ist eine logische NICHT-Verknüpfung. Beispiel: IF A=NOT B |
| NOTE | weist der ersten Variablen die momentane Sektor- und der zweiten Variablen die Bytenummer zu. Beispiel: NOTE #1,S,B |
| ON ERROR | verzweigt das Programm im Fehlerfall zur angegebenen Zeilennummer. Ausschalten mit ON ERROR GOTO 0. Beispiel: ON ERROR GOTO 32000 |
| ON...GOSUB | verzweigt das Programm in Verbindung mit einer Variablen. Ist diese 1, verzweigt es zur ersten angegebenen Zeilennummer; ist sie 2, zur zweiten Zeilennummer usw. Beispiel: ON A GOSUB 100,200,300 usw. |

ATARINSIDE

| PROGRAMMIER WORT | FUNKTION |
|---|---|
| ON...GOTO | ist wie die Anweisung ON...GOSUB, nur daß hier eine Verzweigung mit GOTO stattfindet. Beispiel: ON INDEX GOTO 100,230,400 usw. |
| OPEN | öffnet einen Datenkanal. Es muß mit der Anweisung der gewünschten Übertragungsart ergänzt werden (INPUT, OUTPUT, UPDATE oder APPEND). Beispiel: OPEN #1, "D:DATEI.1" OUTPUT |
| OPTION BASE | bestimmt ob das Variablenfeld mit 1 oder 0 beginnt. Beispiel: OPTION BASE 1. |
| OPTION CHR | reserviert Speicherplatz für einen eigenen Zeichensatz. Beispiel: OPTION CHR1 |
| OPTION PLM | reserviert Speicherplatz für PLAYER-MISSILE-Grafik. Beispiel: OPTION PLM |
| OPTION RESERVE | reserviert Speicherplatz in der genannten Anzahl für Ihre eigenen Anwendungen (Maschinenroutinen etc.) |
| OR | ist eine logische ODER-Verknüpfung. Beispiel: IF A=30 OR B=30 THEN END |
| PEEK | gibt den Inhalt einer bestimmten Speicherzelle als Dezimalzahl aus. Beispiel: A=PEEK(&600) |
| PLM | reserviert in Verbindung mit der Anweisung OPTION Speicherplatz fuer die PLAYER-MISSILE-Grafik. PLM1 = eine Zeile Auflösung, PLM2 = zwei Zeilen Auflösung, PLM0 löscht die Reservierung. Beispiel: OPTION PLM 1 |
| PLOT | zeichnet einen Punkt auf den Bildschirm oder zeichnet eine Linie von einem Plotpunkt zum anderen. Beispiel: PLOT 10,10 TO 20,20 |
| POKE | gibt einer bestimmten Speicherzelle einen neuen Inhalt. Beispiel: POKE &600,255 |
| PRINT | gibt Daten an bestimmte Datenkanäle aus (Bildschirm, Diskette oder Programmrekorder). Beispiel: PRINT USING "!";A$,B$ |



| PROGRAMMIER WORT | FUNKTION |
|---|---|
| PUT | schreibt Byte-weise Daten an den Datenkanal. Beispiel: PUT #1,A |
| RANDOMIZE | bestimmt eine Zufallszahlenfolge. Beispiel: RANDOMIZE oder RANDOMIZE 64 |
| READ | liest den nächsten Wert aus einer DATA-Zeile. Beispiel: READ A,A$ |
| REM | erlaubt es, in das Programm Ihre Kommentare und Notizen in eine Zeile zu schreiben, um die Programmierung zu erleichtern. Sie werden bei der Programmausführung nicht beachtet. Es kann auch durch ! oder ' gesetzt werden. Beispiel: REM hier steht Variable ASX |
| RENUM | erlaubt das Neunumerieren bestehender Programme oder Programmteile. Beispiel: RENUM 100,100 |
| RESERVE | wird in Verbindung mit der Anweisung OPTION verwendet und reserviert eine Anzahl Bytes im Speicher. Beispiel: OPTION RESERVE 1200 |
| RESTORE | setzt DATA-Zeiger zurück. Das Einlesen der Daten beginnt wieder mit dem ersten Wert. Beispiel: RESTORE 100 |
| RESUME | wird bei der Verwendung der Anweisung ON ERROR benötigt, um dem Computer mitzuteilen, wo er das Programm im Fehlerfall fortsetzen soll. Beispiel: RESUME |
| RETURN | setzt nach Abarbeiten der Unterroutine die Programmausführung mit der nach dem GOSUB folgenden Anweisung fort. Beispiel: RETURN |
| RIGHT$ | gibt die letzten Zeichen eines Strings aus. Beispiel: A$=RIGHT$(X$,8) |
| RND | generiert eine Zufallszahl. Ist der angegebene Parameter 0, wird eine Zahl zwischen 0 und 1 ausgegeben. Ist der Parameter > 1 liegt die ausgegebene Zahl zwischen 0 und diesem Parameter. Beispiel: E=RND(10) |
| RUN | startet die Programmausführung in der niedrigsten Zeilennummer. Beispiel: RUN |

ATARINSIDE

| PROGRAMMIER WORT | FUNKTION |
|---|---|
| SAVE | speichert Programm im Speicher unter angegebenem Namen ab.<br>Beispiel: SAVE"D:TEST.1" |
| SCRN$ | liest ein Zeichen vom Bildschirm oder gibt den Farbwert eines Farbpunktes an.<br>Beispiel: A$=ASC(SCRN$(25,2)) |
| SETCOLOR | bestimmt die Farbe und Helligkeit eines bestimmten Farbregisters.<br>Besipiel: SETCOLOR 0,5,5 |
| SGN | gibt an, ob eine Variable positiv, negativ oder Null ist.<br>1 wenn Variable >0 ist,<br>0 wenn Variable =0 ist,<br>-1 wenn Variable <0 ist.<br>Beispiel: B=SGN(Y) |
| SIN | gibt den Wert des Ausdrucks des Sinus in Radiant an.<br>Beispiel: B=SIN(X) |
| SOUND | initialisiert einen der Tongeneratoren.<br>Beispiel: SOUND 1,121,8,10,60 |
| SPC | wird bei der PRINT-Anweisung verwendet, um Leerstellen einzufügen.<br>Beispiel: PRINT SPC(5),A$ |
| SQR | errechnet die Quadratwurzel eines Wertes.<br>Beispiel: A=SQR(D) |
| STACK | gibt den Wert des Zeitzählers an.<br>Beispiel: A=STACK |
| STATUS | gibt den Status des gewünschten Datenkanals an.<br>Beispiel: ST=STATUS(2) |
| STOP | unterbricht die Programmausführung. Die Werte aller Variablen bleiben erhalten.<br>Beispiel: STOP |
| STR$ | wandelt eine numerische Variable in einen String um.<br>Beispiel: A$=STR$(A) |
| STRING$ | definiert einen String auf zwei Arten:<br>1. wiederholt das angegebene Zeichen (im Beispiel 100 mal den Buchstaben A).<br>Beispiel: X$=STRING$(100,"A")<br><br>2. wiederholt das dem ASCII-Code entprechende Zeichen (im Beispiel wieder 100 mal A)<br>Beispiel: Y$=STRING$(100,65) |



| PROGRAMMIER WORT | FUNKTION |
|---|---|
| TAB | wird in Verbindung mit der PRINT-Anweisung als Tabulator verwendet.<br>Beispiel: PRINT TAB(20),A$ |
| TAN | gibt den Tangens eines arithmetischen Ausdrucks in Radiant an.<br>Beispiel: D=TAN(3.14) |
| TIME | gibt den Wert der internen Uhr an.<br>Beispiel: A=TIME |
| TIME$ | gibt die Zeit in 24 Stunden-Format an (STD:MIN:SEK).<br>Beispiel: TIME$="08:55:05"<br>PRINT TIME$ |
| TROFF | schaltet Trace-Funktion aus.<br>Beispiel: TROFF |
| TRON | schaltet Trace-Funktion ein.<br>Beispiel: TRON |
| UNLOCK | hebt die Anweisung LOCK wieder auf.<br>Beispiel: UNLOCK"D:TEST.1" |
| USING | gibt Variablen in einem bestimmten Format aus.<br>Beispiel: PRINT USING "###.##";DM |
| USR | ruft Maschinenspracheunterroutinen auf.<br>Beispiel: X=USR(ADDR(VARPTR(A(0))) |
| VAL | wandelt die Ziffern eines Strings in eine numerische Variable um.<br>Beispiel: A=VAL(A$) |
| VARPTR | gibt die Startadresse des durch OPTION reservierten Speicherbereichs an.<br>Beispiel: ADDR=VARPTR(RESERVE) |
| VERIFY | vergleicht das aufgerufene Programm mit dem noch im Speicher vorhandenen Programm.<br>Beispiel: VERIFY "D:PROGRAM.BAS" |
| WAIT | wartet, bis ein bestimmter Parameter den angegebenen Wert erreicht hat.<br>Beispiel: WAIT &E456,&FF,30 |
| XOR | ist eine logische exklusiv-Oder-Verknüpfung.<br>Beispiel: IF A XOR B=0 THEN END |

ATARINSIDE

## Anlage O BEDEUTUNG DER FEHLERMITTEILUNGEN

| CODE | BILDSCHIRM TEXT | FEHLER |
|---|---|---|
| 1 | NEXT WITHOUT FOR ERROR IN Zeile#, | Es wurde die NEXT-Anweisung benutzt, ohne die erforderliche FOR-Anweisung zu geben. |
| 2 | SYNTAX ERROR IN Zeile# | Syntaxfehler. Eine Anweisung wurde falsch eingegeben, die Interpunktion ist falsch oder es bestehen noch offene Klammern. |
| 3 | RETURN WITHOUT GOSUB ERROR | Es wurde ein RETURN ohne die dazugehörige GOSUB-Anweisung eingegeben. |
| 4 | OUT OF DATA ERROR IN Zeile# | Für die Ausführung einer INPUT- oder READ-Anweisung sind nicht genügend Daten vorhanden. |
| 5 | FUNCTION CALL ERROR IN Zeile# | Es wurde versucht, eine Funktion unter Benutzung illegaler Parameter auszuführen, z.B. die Quadratwurzel einer negativen Zahl zu ziehen. |
| 6 | OVERFLOW | Es ist eine zu große oder zu kleine Zahl als Ergebnis einer Berechnung oder Eingabe entstanden. |
| 7 | OUT OF MEMORY ERROR | Der zur Verfügung stehende Speicherplatz ist belegt. |
| 8 | UNDEF'D LINE ERROR IN Zeile# | Programmverzweigung zu einer nicht existierenden Zeilennummer. |
| 9 | SUBSCRIPT ERROR IN Zeile# | Ein Element der Matrix wurde über den dimensionierten Bereich hinaus aufgerufen. |
| 10 | REDEF'N ERROR IN Zeile# | Es wurde versucht, eine bereits dimensionierte Variable erneut zu dimensionieren. |
| 11 | DIVISION BY ZERO | Es versucht versucht durch 0 zu teilen. |
| 12 | ILLEGAL DIRCT ERROR | Eine der Anweisungen INPUT, GET oder DEF wurde direkt eingegeben (ohne Zeilennummer). |



| CODE | BILDSCHIRM TEXT | FEHLER |
|---|---|---|
| 13 | TYPE MISMATCH ERROR IN Zeile# | Es wurde versucht einer numerischen Variablen einen alphanumerischen Wert zuzuweisen oder umgekehrt. |
| 14 | FILE I/O ERROR | Genereller Ein- oder Ausgabe-Fehler. |
| 15 | QUANTITY TOO BIG ERROR IN Zeile# | String-Variablen dürfen nur 255 Zeichen lang sein. |
| 16 | FORMULA TOO COMPLEX ERROR | Eine mathematische oder String-Funktion ist zu komplex. Sie muß in kleinere Schritte zerlegt werden. |
| 17 | CAN'T CONTINUE ERROR | Ein CONT-Befehl kann nicht ausgeführt werden, weil das Programm bereits beendet ist. |
| 18 | UNDEF'D FUNCTION ERROR | Die USR-Funktion kann nicht ausgeführt werden, da sie einen logischen Fehler enthält oder weil die angegebene Startadresse nicht richtig ist. |
| 19 | NO RESUME ERROR IN Zeile# | Das Ende eines Programms wurde nach einem ON ERROR erreicht, ohne das eine RESUME-Anweisung gegeben wurde. |
| 20 | RESUME WITHOUT ERROR IN Zeile# | Eine RESUME-Anweisung wurde vor einem ON ERROR eingegeben. |
| 21 | FOR WITHOUT NEXT ERROR | Ein FOR wurde ohne die zugehörige NEXT-Anweisung gegeben. |
| 22 | LOCK ERROR | Es wurde versucht ein mit SAVE...,LOCK abgespeichertes Programm zu listen oder zu ändern. |
| 23 | TIME ERROR | Es wurde versucht, nach einem AFTER keine RESUME-Anweisung zu geben. |

ATARINSIDE

FEHLER IM ZUSAMMENHANG MIT EIN- UND AUSGABEFUNKTIONEN

| HEX | DEZ | BEDEUTUNG |
|---|---|---|
| 80 | 128 | BREAK-Taste wurde gedrückt |
| 81 | 129 | der gewählte Datenkanal ist bereits geöffnet |
| 82 | 130 | das gewählte Gerät existiert nicht |
| 83 | 131 | Datenkanal ist nur zum Schreiben geöffnet |
| 84 | 132 | unvollständige / falsche Anweisung |
| 85 | 133 | Datenkanal / Datei nicht geöffnet |
| 86 | 134 | falsche Datenkanal-Nummer |
| 87 | 135 | Datenkanal ist nur zum Lesen geöffnet |
| 88 | 136 | Versuch über das Dateiende zu lesen |
| 89 | 137 | Datei abgeschnitten |
| 8A | 138 | Gerät antwortet nach einiger Zeit nicht |
| 8B | 139 | Negatives / nicht erfolgtes Quittungssignal vom Gerät |
| 8C | 140 | Lesefehler auf seriellem Bus |
| 8D | 141 | Cursor befindet sich außerhalb des Bildschirms |
| 8E | 142 | Computer verarbeitet eintreffende Daten zu langsam |
| 8F | 143 | Prüfsummen Fehler |
| 90 | 144 | Gerätefehler |
| 91 | 145 | Unzulässiger Bildschirm-Modus |
| 92 | 146 | dem Datenkanal wurde eine unbekannte Anweisung gegeben |
| 93 | 147 | für den gewählten Grafikmodus zu wenig Speicherplatz |
| A0 | 160 | nicht vorhandene Diskettenstation angesprochen |
| A1 | 161 | zu viele offene Diskettendateien |
| A2 | 162 | Diskette ist vollgeschrieben |
| A3 | 163 | grober Lese-/Schreibfehler |
| A4 | 164 | Dateinummern / Sektoren vertauscht |
| A5 | 165 | falscher Dateiname |
| A6 | 166 | unzulässiger POINT-Befehl |
| A7 | 167 | Datei blockiert (LOCK-Befehl) |
| A8 | 168 | unzulässiger Diskettenbefehl |
| A9 | 169 | Direktory dieser Diskette ist voll (64 Namen) |
| AA | 170 | gewählte Datei befindet sich nicht auf der Diskette |
| AB | 171 | unzulässiger POINT-Befehl |

ATARINSIDE

# Wichtige Informationen

Lieber Computerfreund, lieber Kunde, lieber Händler!

Jeder, der sich einmal selbst damit beschäftigt hat, ein Computerprogramm zu fertigen, weiß, welche Arbeit und geistige Mühe aufgewendet werden muß, um eine Problemlösung zu finden und sie anwenderfreundlich zu programmieren. Die Erfüllung dieser Voraussetzungen erfordert viel Erfahrung und hohe finanzielle und zeitliche Investitionen. Das Ergebnis sind gute und erfolgreiche Computerprogramme, die von interessierten Anwendern nachgefragt werden und deshalb für den Händler verkäuflich sind.

Diese Tatsache machen sich einige dadurch zunutze, daß sie die mit hohen Voraufwendungen geschaffenen erfolgreichen Programme der Firma Atari kopieren oder ihren Kunden die Möglichkeit anbieten, die gewünschten Programme auf Diskette zu überspielen. Sie meinen, damit ihren Kunden ein gutes und billiges Angebot zu machen. Die Kunden wissen jedoch meist nicht, daß sie lediglich ein vermeintlich gutes und billiges Angebot erhalten.

Abgesehen davon, daß das Angebot zur Überspielung von Programmen und das Anbieten und Verkaufen illegal kopierter Programme strafrechtlich verboten ist, weil es sich dabei um Verletzungen des Urheberrechtes (COMPUTERPROGRAMM PIRATERIE) handelt, die von Atari gegenüber jedermann ohne Ansehen der Person gerichtlich verfolgt wird, so ist auch die Annahme falsch, das Angebot sei günstig oder billig:

- Gestohlene Ware ist immer billig. Der Dieb hat keine Voraufwendungen. Er eignet sich nur fremdes Eigentum an, für die der Käufer keine Gewährleistung erhält.
- Der Händler, der das Kopieren von Programmen anbietet, anstatt Originale zu verkaufen schmarotzt an fremder Leistung.
- Der interessierte Kunde wird bald keine guten Programme mehr kaufen können und illegale Programme wird der Handel bald auch nicht mehr anbieten können.

Letzteres deswegen, weil niemand mehr bereit und in der Lage sein wird, gute verkaufsfähige Programme zu entwickeln, wenn nicht die Möglichkeit besteht, die hohen Voraufwendungen durch Verkäufe wieder zu verdienen. Die Piraten sind geistig weder in der Lage noch überhaupt bereit, sich der Mühe zu unterziehen, Programme zu entwickeln. Sie können und wollen nur durch Diebstahl fremder guter Leistung eine schnelle bequeme Mark verdienen.

Wer also Interesse daran hat, daß das Angebot an guten Computerprogrammen wächst, sollte die illegalen „billigen" Angebote meiden und mit dazu beitragen, daß den Totengräbern der Computer-Programmentwicklung und damit des Computerhandels das Handwerk gelegt wird.

Wir danken für Ihr Verständnis und freuen uns über jeden Hinweis von Ihnen.

Atari Elektronikvertriebsges. mbH

ATARI ELEKTRONIK Vertriebsgesellschaft mbH · Bebelallee 10 · 2000 Hamburg 60 · Tel. 040/511 80 91

ATARINSIDE

ATARI®

A Warner Communications Company

ATARI-Elektronik Vertriebsgesellschaft mbH
Postfach 60 01 69 · Bebelallee 10 · 2000 Hamburg 60

Jegliche Rechte vorbehalten.
Vermietung, Verleih, Vervielfältigung
und öffentliche Aufführung verboten.

ATARINSIDE