

ATARI BASIC

REFERENCE GUIDE

LEITFADEN

MANUAL DE REFERENCIA

MANUALE D'USO

MANUEL DE REFERENCE

HANDLEIDING

ATARI[®]

FÜR ERFAHRENE PROGRAMMIERER

Das Lernen von BASIC ist wie das Lernen irgendeiner Sprache — es kostet ein bißchen Zeit und Mühe, aber im Endeffekt lohnt es sich sehr. Dieses Handbuch gibt Informationen über ATARI BASIC — einen beliebten, leistungsstarken Dialekt für Programmierer, die schon mit der BASIC-Programmiersprache vertraut sind. Dieses Handbuch dient nur als Nachschlagewerk. Es beinhaltet weder umfangreiche Programmbeispiele noch Lernmaterial für Anfänger. Anfänger sowie erfahrene Programmierer sollten zwecks weiterer Informationen auf die folgenden Handbücher zurückgreifen: ATARI BASIC von Albrecht, Finkel und Brown; ATARI BASIC REFERENCE MANUAL; und INSIDE ATARI BASIC von Bill Carris.



INDEX

BEFEHLE	SEITENNUMMER	PADDLE	
ABS	19	PEEK	20
ADR	20	PLOT	20
AND	14	POINT	19
ASC	20	POKE	17
ATN	20	POP	18
BEARBEITUNGS-FUNKTIONEN	21	POSITION	16
BYE	15	PRINT	19
CLOAD	15	PTIG	17
CHAR\$	20	PUT	20
CLOSE	19	RAD	17
CLOSE	17	READ	20
CLR	18	REM	18
COLOR	18	RESTORE	18
COM	18	RETURN	16
CONT	15	RND	19
COS	20	RUN	15
CSAVE	15	SAVE	15
DATA	18	SETCOLOR	18
DEG	20	SGN	19
DIM	18	SIN	20
DOS	15	SOUND	15
DRAWTO	19	SPEZIELLE FUNKTIONSTASTEN	21
END	16	SQR	19
ENTER	15	STATUS	17
EXP	19	STICK	20
FEHLERMELDUNGEN	21	STRIG	20
FOR	16	STOP	16
FRE	20	STR\$	20
GET	17	THEN	16
GOSUB	16	TO	16
GOTO	16	TRAP	16
GRAPHIK-BEFEHLE	18	USR	20
IF	16	VAL	20
INPUT	17		
INT	19		
LEN	20		
LET	18		
LIST	15		
LOAD	15		
LOCATE	19		
LOG	19		
LPRINT	17		
NEW	15		
NEXT	16		
NOT	14		
NOTE	17		
ON	16		
OPEN	17		
OPERATOREN	14		
OR	14		



REIHENFOLGE DER OPERATOREN

Die in der innersten Klammer aufgeführten Befehle werden zuerst ausgeführt und dann in die nächste Stufe übertragen. Sind Klammern in anderen Klammern enthalten, so nennt man diese "verschachtelte" Klammern. Befehle in der gleichen Verschachtelungsstufe werden in der folgenden Reihenfolge ausgeführt:

HÖCHSTE PRIORITÄT

<>=<>=<>

Vergleiche innerhalb einer Klammer haben die gleiche Priorität und werden von links nach rechts ausgeführt.

-
Negativer Wert, z. B.: x=-1

Exponentialfunktion.

*/

Multiplikation und Division haben die gleiche Priorität und werden von links nach rechts ausgeführt.

+ -

Addition und Subtraktion haben die gleiche Priorität und werden von links nach rechts ausgeführt.

<>=<>=<>

Vergleiche innerhalb einer Klammer haben die gleiche Priorität und werden von links nach rechts ausgeführt.

NOT

Logische Verknüpfung

AND

Logisches AND

OR

Logisches OR

NIEDRIGSTE PRIORITÄT

(Erlaubte Abkürzungen in Klammern)
ATARI Computer machen keinen Unterschied zwischen einem Befehl (COMMAND) und einer Anweisung (STATEMENT). Die folgenden Worte können im Programm oder direkt verwendet werden.

KONTROLLE DES SYSTEMS

Bye (B.) — Gehst aus dem BASIC in den SELF TEST MODUS über.

DOS — Zeigt das DOS-Menü an (nur bei Verwendung einer Diskettenstation).

CSAVE (CS) — Speichert ein Programm auf Cassette.

CLOAD — Lädt ein Programm von einer Cassette ein.

SAVE (S) — Speichert ein BASIC-Programm auf Diskette.
z.B.: SAVE "D:MYFILE.BAS".

LOAD (LO) — Liest ein Programm von einer Diskette ein.
z.B.: LOAD "D:MYFILE.BAS".

LIST (L) — Schreibt ein Programm auf den Bildschirm oder stellt es einem "Output-Gerät" zur Verfügung.

z.B.: LIST (Schreibt das Programm auf den Bildschirm)

LIST 10 (Schreibt Zeile 10 auf den Bildschirm)

LIST 10, 20 (Schreibt alles von Zeile 10 bis Zeile 20)

LIST "P." (Stellt das Programm dem Drucker zur Verfügung)

LIST "P.", 10, 20 (Stellt Zeilen 10 bis 20 dem Drucker zur Verfügung)

LIST "D:MYFILE.LST" (Stellt das Programm der Diskettenstation zur Verfügung)

LIST "D:MYFILE.LST", 10, 20 (Stellt die Zeilen 10 bis 20 der Diskettenstation zur Verfügung)

LIST "C." (Stellt das Programm dem Programm-Recorder zur Verfügung)

ENTER (E) — Liest das Programm im Tasterformat ein.

z.B.: ENTER "C."

ENTER "D:MYFILE.LST"

Achtung: Ein bereits geladenes Programm wird zeilengleich überschrieben!

NEW — Löscht den Speicher.

RUN — Programmstart in BASIC. Dabei kann das Programm im Speicher vorliegen oder von Diskette oder Cassette eingelesen werden (setzt Variablen auf Null und löscht die Dimensionierung von Variablenfeldern).

z.B.: RUN (führt ein im Speicher befindliches Programm aus)

RUN "D:MYFILE.BAS" (Programm von Diskette laden und starten)

CONT — Führt ein Programm mit der nächsten numerischen Zeile weiter, nachdem die BREAK-Taste gedrückt wurde oder das Programm bis zum STOP oder END ausgeführt worden ist.

TONBEFEHL

SOUND (SO) — Stellt einen der vier Tonkanäle ein. Der Ton wird so lange ausgestrahlt, bis ein neuer SOUND-Befehl für diesen Kanal erfolgt oder ein END-, RUN- oder NEW-Befehl auszuführen ist. Die Kanäle sind unabhängig voneinander zu programmieren und können gleichzeitig aktiviert werden. Dem Befehl müssen vier Werte folgen (Ziffern, numerische Variablen oder mathematische Ausdrücke),

z.B.: SOUND A, B, C, D

A = Tonkanal-Nummer (0-3)

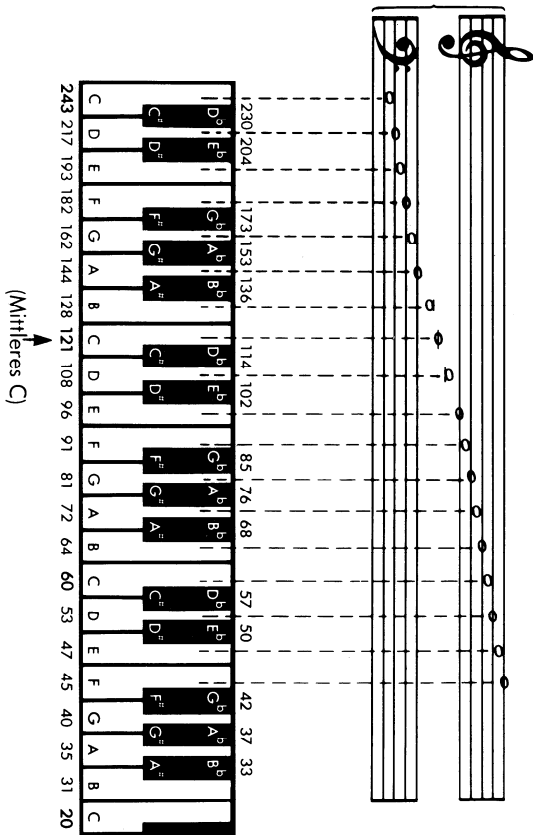
B = Tonhöhe (0-255). Je größer der Wert, desto tiefer die Frequenz.

Frequenz = 31960 / (Tonhöhe + 1) siehe Tabelle.

C = Verzerrung (0-14). Nur gerade Zahlen. Nur 4 und 10 sind "reine" Töne.

D = Lautstärke (0-15). Je größer der Wert, desto größer die Lautstärke. 0 bedeutet kein Ton. Ist die Gesamtlautstärke aller vier Tonkanäle größer als 32, fängt der Lautsprecher evtl. an zu brummen.

BEZIEHUNG DER KLAVIERTASTEN ZU DER TONLEITER



KONTROLLE DES PROGRAMMS

GOTO (G.) — Die Ausführung des Programms wird bei der genannten Zeilennummer weiter geführt,
z.B.: GOTO 30 (führt das Programm ab Zeile 30 aus)

GOTO A + 10 (nimmt Inhalt der Variable A, addiert 10 und springt auf diese neue Zeilennummer)

ON ... GOTO — Die Ausführung des Programms wird bei der durch einen Ausdruck definierten Zeile weitergeführt,
z.B.: ON A GOTO 10, 300, 50

(IF A = 1 THEN GOTO 10;
IF A = 2 THEN GOTO 300;
IF A = 3 THEN GOTO 50)

GOSUB (GOS.) — Springt auf ein Unterprogramm mit definierter Zeilennummer,
z.B.: ON A+1 GOSUB 10, 300, 50

(IF A+1 = 1 THEN GOSUB 10;
IF A+1 = 2 THEN GOSUB 300;
IF A+1 = 3 THEN GOSUB 50)

RETURN (RET) — Beendet ein Unterprogramm und springt auf das nächste der GOSUB-Anweisung folgende Statement.

Vorsicht: Wert der Ausdruck eine Zahl aus, die kleiner als 1 oder größer als die Zahl der Zeilennummern aus, sind die Resultate nicht vorhersehbar.

FOR (F) — Dieser Befehl definiert die Anfangs- und Endwerte einer Indexvariablen sowie den Wert, der dieser jedesmal dann hinzugezählt werden muß, wenn eine FOR ... NEXT-Schleife ausgeführt wird. Der hinzugezählte Wert ist immer 1, außer er wird durch einen STEP-Befehl anderweitig bestimmt. Ein NEXT-Befehl bewirkt, daß die Statements zwischen FOR und NEXT wiederholt werden,
z.B.: FOR A = 1 TO 10 (A wird, bei 1 beginnend, um 1 erhöht und bei 10 anhalten)

FOR A = 10 TO 1 STEP -2 (A wird, bei 10 beginnend, um 2 vermindert und stoppt bei 2)

FOR A = B/T TO B*T STEP X

NEXT (N.) — Beendet eine FOR ...

NEXT-Schleife. Kontrolliert, daß der Indexwert nicht größer als der Endwert geworden ist, erhöht den Index-Wert um den STEP-Wert und führt die Ausführung des Programms beim Statement nach dem FOR weiter. Falls der INDEX den END-Wert überschreitet, wird zum Statement nach dem NEXT-Befehl zurückgegangen,
z.B.: NEXT A (A ist die Index-Variable, die in dem For-Statement definiert wurde).

POP — Löscht RETURN-Befehle im Zusammenhang mit dem zuletzt ausgeführten GOSUB-Befehl,

z.B.: POP: GOTO 10 (dient zum vorzeitigen Verlassen eines Unterprogramms ohne Ausführung des RETURN-Befehls)

IF..THEN — Das Statement nach THEN wird dann ausgeführt, wenn die Bedingung zwischen IF und THEN 'wahr' ist. Sonst wird zur nächsten Programmzeile übergegangen,

z.B.: IF A = B THEN GOTO 300

IF A ≠ B THEN PRINT "A = B": PRINT

"HOW ABOUT THAT!":

LET A = 5: GOTO 20

IF A THEN PRINT "A is non-zero"
(Der Ausdruck 'A' ist nicht gleich Null' ist wahr).

TRAP (T) — Mit einem TRAP-Statement können Anweisungen für den Fall einer Fehlermeldung gegeben werden,

z.B.: TRAP 30 (bei einem Fehler erfolgt Sprung zu Zeile 30)

TRAP 40,000 (Zeilennummern, die größer als 32767 sind, schalten TRAP aus.)

STOP — Stoppt das Programm und drückt die entsprechende Zeilennummer aus. Schließt die Datei und die Tonkölle nicht. Programm kann mit CONT wieder begonnen werden.

z.B.: IF A = B THEN STOP

END — Beendet das Programm und schließt alle offenen Dateien und Tonkanäle. Das Programm kann mit CONT erneut gestartet werden.

EINGABE UND AUSGABE

GERÄTENAMEN

Für den systeminternen Datenaustausch wird jedes Gerät mit einer speziellen Bezeichnung angesprochen. Die

Disketten-Station und das Interface-Modul (RS 232 Handler) haben zusätzlich eine Ansprechnummer (1-4). Die Diskettenstation benötigt darüberhinaus noch einen Dateinamen. Dieser muß in "..." oder in String-Variablen enthalten sein. Nachfolgend einige Beispiele:

K: Tastatur. Nur für Eingabe

P: Drucker. Nur für Ausgabe

C: Kassette. Eingabe und Ausgabe

S: Bildschirm (TV). Nur Ausgabe

E: Bildschirm — Bearbeitungsgerät (Tastatur kombiniert mit Bildschirm).

Eingabe und Ausgabe

R: RS232 Handler (A/ARI Interface

Modul) Eingabe und Ausgabe

D: DATEINAME. ERWEITERUNG

(3 Ziffern) der Datei, z.B.:

"D:FILENAME.EXT"

Diskettenstation 1

D2: DATEINAME. ERWEITERUNG

(gleiche Datei in Diskettenstation 2)

LEITFADEN

EINGABE-/AUSGABE-BEFEHLE (I/O)

Dateinamen können bis zu acht Buchstaben/Ziffern lang sein und müssen mit einem Buchstaben beginnen. Der Name der Datei kann mit einer wohlweisen Erweiterung enden (ein Punkt, gefolgt von 1 - 3 beliebigen Buchstaben/Ziffern). Einige praktische Erweiterungen sind:

- .BAS = Mit der Funktion "SAVE" versehenes BASIC-Programm.
- .LIST = Mit der Funktion "LIST" versehenes BASIC-Programm.
- .DAT = Allgemeine Daten-Dateien.
- .OBJ = Maschinensprache-Dateien ('object files').
- .TXT = Text-Dateien.

OPEN (O) — Bereitet ein Gerät für die Ein- oder Ausgabe vor. Die IOCB-Nummern sind 1-7. Verwendet folgende Codes:

TYP	CODE	OPERATIONEN
Eingabe	4	Nur einlesen
Ausgabe	8	Nur schreiben
Update	12	Einlesen und schreiben
Append	9	Zusatz zum Dateiende
Directory	6	Nur Inhalt der Diskette

z.B.: OPEN #1, 4, 0, "K:" (Vorbereitung der Tastatur zur Eingabe bei IOCB #1)

OPEN #2, 8, 0, "P:" (Vorbereitung des Druckers für Ausgabe bei IOCB #2)

OPEN #1, 12, 0, "D:\MYFILE.DAT" (Vorbereitung der Datei MYFILE.DAT für "Update"-Funktionen IOCB #1)

OPEN #1, 6, 0, "D:.*" (Vorbereitung des Disketteninhalts für die Adresse bei IOCB #1)

z.B.: CLOSE #1 (Schließt die bei IOCB #1 vorbereitete Datei und gibt IOCB frei.)

INPUT (I) — Liest eine Zeile von Daten aus dem Gerät ein. Diese Zeile muß durch ein RETURN-Zeichen beendet werden, z.B.: INPUT A (Tastatureingabe einer Zahl, die in Variable A gesetzt wird)

INPUT A, B, C (Tastatureingabe von drei durch Komma zu trennenden Zahlen)

INPUT A\$ (Tastatureingabe von Daten, die in Variable A\$ gesetzt werden. Diese Eingabe ist durch RETURN-Taste abzuschließen.)

INPUT #1, A (Eingabe einer Zahl von einem bei IOCB #1 vorbereiteten Gerät in Variable A)

PRINT (PR) oder (?) — Läßt die Daten auf dem Bildschirm oder einem anderen durch OPEN vorbereiteten Gerät erscheinen

z.B.: PRINT (eine leere Zeile erscheint)

PRINT "die Zahl ist", A (das Komma bewirkt Leerstellen zwischen Text und Zahl, POKE 201,5 bewirkt 5 Leerstellen.)

PRINT A\$; ";" bewirkt die Darstellung des folgenden PRINT-Befehls in derselben Zeile)

PRINT #1, A\$ (Ausdruck des Inhalts von Variable A\$ durch das bei IOCB #1 vorbereitete Gerät).

LPRINT (LP) — Druckt ohne OPEN-Befehl auf dem Drucker aus,

z.B.: LPRINT (eine leere Zeile wird "gedruckt")

LPRINT A\$ (Inhalt der Variable A\$ wird ausgedruckt)

LPRINT A\$,B (der Inhalt von Variable A\$ und B wird in der selben Zeile dargestellt)

GET — Holt ein einzelnes Byte aus dem bestimmten Gerät und setzt es in die definierte Variable ein,

z.B.: GET #1, A (holt ein Byte aus dem bei IOCB #1 vorbereiteten Gerät und setzt es in die numerische Variable ein)

PUT — Setzt das einzelne Byte aus einer numerischen Variable in ein definiertes Gerät ein,

z.B.: PUT #1, A (setzt den Inhalt von Variable A in ein bei IOCB #1 vorbereitetes Gerät ein)

NOTE (NO) — Wird bei Disketten verwendet zur Bestimmung des Ortes des nächsten einzulesenden oder auszuscheidenden Bytes,

z.B.: NOTE #1, SEC, BYTE (setzt die derzeitigen Nummern von Sektor und Byte in SEC bzw. BYTE ein. Bezieht sich dabei auf die bei IOCB #1 vorbereitete Datei)

POINT (P) — Wird verwendet, um DOS mitzuteilen, wo sich das nächste einzulesende oder auszuscheidende Byte befindet,

z.B.: POINT #1, SEC, BYTE (für SEC und BYTE sind die Werte einzusetzen).

STATUS (ST) — Stellt den Status eines Gerätes fest. Die Status-Codes sind der Tabelle 'Fehlermeldung' zu entnehmen,

z.B.: STATUS #1, A (liest den Status für das bei IOCB #1 vorbereitete Gerät und setzt ihn in die Variable A ein)

VERARBEITUNGSBEFEHLE

LET — Weist den Variablen Werte zu,
z.B.: LET A = B (der Wert von Variable B
wird auch in A übernommen)
LET A\$ = "HALLO"
A = B, A\$ = "HALLO" (LET kann
ausgelassen werden)

POKE — Setzt ganze Zahlen zwischen 0
und 255 in eine zu bestimmende Stelle des
Speichers ein. PEEK wird entsprechend
zum Lesen der Speicherstelle verwendet.
z.B.: POKE 82,0 (setzt 0 in die Speicher-
stelle 82 ein)
A = PEEK (82) (liest den Inhalt von
Speicherstelle 82 und setzt ihn in Vari-
able A ein)

DIM — Reserviert im Speicher Platz für
String- und numerische Variablenfelder.
Jeder Zeichenzwischenraum, der für eine
Datenkette reserviert wurde, braucht ein
Byte; jedes Element in einem numerischen
Variablenfeld sechs Bytes,
z.B.: DIM A\$ (10) (eine String-Variablen mit
einer Länge von 10 Bytes)
DIM B (10) (Ein numerisches Varia-
blenfeld; B umfaßt die Elemente 0
bis 10)
DIM B (10, 10) (ein 2-dimensionales
Variablenfeld)
DIM A\$ (10), B (10) (verschiedene
Variablenfelder sind durch Komma
zu trennen)

COM — Entspricht DIM

CLR — Löscht die Dimensionierung aller
Variablenfelder und Datenketten.

DATA (D) — Stellt eine Liste der durch die
unten erwähnte READ-Funktion zu ver-
wendenden Zahlen und/oder Strings auf,
z.B.: DATA 1, 2, 3, 4, A, B, C, D (eine
Datenliste, die mit dem READ-Befehl
gelesen werden soll)

READ — Liest einzeln die nächste Position
in einer DATA-Anweisung und weist sie
einer Variablen zu.

z.B.: READ A (die nächste Zahl der DATA-
Anweisung wird in Variable A
eingesetzt)
READ A\$ (gilt auch für
String-Variablen)
READ A, A\$, B, B\$ (verschiedene
Variablenfelder sind durch Komma
zu trennen)

RESTORE (RES) — Weist hin auf READ in
einer DATA-Anweisung,
z.B.: RESTORE (Das nächste Byte wird die
erste Position der ersten DATA-
Anweisung darstellen)
RESTORE 10 (Das nächste Byte wird
die erste Position der DATA-
Anweisung in Zeile 10 darstellen)

REM (R) oder () — Erlaubt Anmerkun-
gen. Alles, was von REM bis zum Ende der
Zeile folgt, wird von BASIC nicht
beachtet,
z.B.: REM Dies ist eine Bemerkung!

GRAPHIK-BEFEHLE

GRAPHICS (GR) — Wählt den
GRAPHICS-Modus; Modus +16 ergibt
einen ungeteilten Bildschirm (ohne Text-
fenster); Modus +32 löscht den Bildschirm
nicht.

z.B.: GRAPHICS 8 (Graphik-Modus 8 mit
Textfenster)
GRAPHICS 8 +16 (Modus 8, ungeteil-
ter Bildschirm)
GRAPHICS 8 +32 (Bildschirm wird
nicht gelöscht)
GRAPHICS 8 +16 + 32 (Beide
Möglichkeiten kombiniert)

SETCOLOR (SE) — Bestimmt den Farbton
und die Helligkeit des gewählten Farb-
gisters. Die Registernummer ist nicht die-
selbe wie beim COLOR-Befehl.

z.B.: SETCOLOR 1, 2, 4 (Stellt Register 1
auf Farbton 2 und Helligkeit 4)
Register (0-4)
Farbtöne (0-15)
Helligkeiten (0-14, nur gerade
Zahlen).

COLOR (C) — Aus den Graphik-Modi
(3-11) wird ein Farbregister für die Ver-
wendung im PLOT-Befehl gewählt. Das
Register entspricht hier nicht demjenigen
in SETCOLOR,
z.B.: COLOR 2 (wählt Farbregister 2)

TABELLE DER GRAPHIK-MODI UND
BILDSCHIRMFORMATE

BILDSCHIRMFORMATE									
"Graphics"- -Modus	Modus/ Typ	Spalten	Zeilen/ Geteilter Bildschirm	Zeilen/ Ungeteilter Bildschirm	Anzahl Farben	Erfordertes RAM (Bytes) geteilt ungeteilt			
0	TEXT	40	—	24	1-1/2	672	992		
1	TEXT	20	20	24	5	672	672		
2	TEXT	20	10	12	5	424	420		
3	GRAPHICS	40	20	24	4	434	432		
4	GRAPHICS	80	40	48	2	694	696		
5	GRAPHICS	80	40	48	4	1174	1176		
6	GRAPHICS	160	80	96	2	2174	2184		
7	GRAPHICS	160	80	96	4	4190	4200		
8	GRAPHICS	320	160	192	1-1/2	8112	8138		
9	GRAPHICS	80	—	192	1	—	8138		
10	GRAPHICS	80	—	192	9	—	8138		
11	GRAPHICS	80	—	192	16	—	8138		
12	GRAPHICS	40	20	24	5	1154	1152		
13	GRAPHICS	40	10	12	5	664	660		
14	GRAPHICS	160	160	192	2	4270	4296		
15	GRAPHICS	160	160	192	4	8112	8138		

FUNKTIONEN

PLOT (PL.) — Setzt einen einzelnen Punkt

oder ein Zeichen an eine bestimmte Stelle auf dem Bildschirm,

z.B.: PLOT X, Y (XY Koordinaten, nur positive Werte)

POSITION (POS.) — Wählt eine Bild-

schirmposition, es wird jedoch nicht auf-
gezeichnet. Dies ist für die Positionierung
des Textes mit PRINT nützlich,

z.B.: POSITION X, L

LOCATE (LOC.) — Findet Daten, die an

einer spezifischen Stelle auf dem Bild-
schirm gespeichert sind. Verwendet in den
Modi 0-2 Zeichen und in den Modi 3-11

Farbwerte,

z.B.: LOCATE X, Y, D (Geht nach X, Y und
legt die Daten in D ab)

DRAWTO (DR.) — Zieht eine Linie zwi-
schen der letzten Position des Cursors
(Markierungszeichen) und der spezifi-
zierten X-Y Koordinate. Der Cursor springt
zur Zielkoordinate,

z.B.: DRAWTO X - Y (zieht eine Linie zu
Punkt X, Y von der jetzigen Position
des Cursors aus).

Eine Funktion ist eine veränderliche
Größe, die in ihrem Wert von einer an-
deren abhängig ist. Werte können Strings,
Zahlen oder mathematische Ausdrücke
sein. Die Beispiele zeigen A (eine nume-
rische Variable) oder A\$ (eine String-
variable), die gleich der Funktion sind;
eine Funktion kann jedoch fast überall
dort gebraucht werden, wo man einen
Wert verwenden würde, einschließlich in
einer anderen Funktion.

ARITHMETISCHE FUNKTIONEN

ABS — Gibt den absoluten ("unsigned")

Wert einer Zahl,

z.B.: A = ABS (B)

CLOG — Gibt den Logarithmus zur Basis
10,

z.B.: A = CLOG (B)

EXP — Gibt den Wert um den spezifi-
zierten Faktor potenzierten von 'e'. Diese

Funktion ist reziprok zur LOG Funktion,
z.B.: A = EXP (B)

INT — Gibt den ganzzahligen Wert eines
Ausdrucks,

z.B.: A = INT (B)

LOG — Gibt den natürlichen Logarithmus
eines Wertes. Diese Funktion ist reziprok
zur EXP-Funktion.

RND — Gibt eine Zufallszahl, die gleich
oder größer '0' und kleiner als '1' ist (der
Wert hat keinen Einfluss),

z.B.: A = RND (0)

A = RND (0)*8 (A = eine Zahl, die
gleich oder größer als '0' und kleiner
als '8' ist)

SGN — Gibt eine -1, falls der Wert
negativ ist, eine 0, falls der Wert null ist
und eine 1, falls er positiv ist,

z.B.: A = SGN (A).

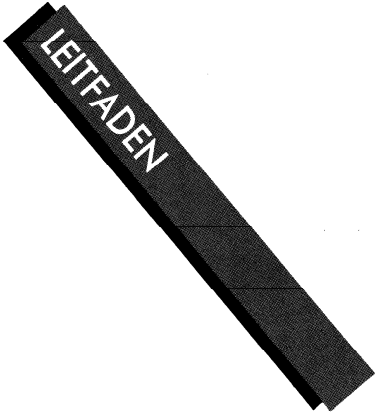
SQR — Gibt die positive Quadratwurzel
eines positiven Wertes,

z.B.: A = SQR (B).

DIE ATARI-FARBEN (PAL)

Farben (SETCOLOR)	Werte (SETCOLOR)
Hellgrau	0
Goldgelb	1
Hellorange	2
Pink	3
Lila	4
Flieder	5
Hellblau	6
Blaurot	7
Himmelblau	8
Azurblau	9
Türkis	10
Grasgrün	11
Grün	12
Gelbgrün	13
Oliv	14
Ocker	15

SETCOLOR (Farben- register)	Fehlfarben- Code	Helligkeit	Richtige Farbe
0	2	8	Hellorange
1	12	10	Grün
2	9	4	Blau
3	4	6	Lila
4	0	0	Schwarz



TRIGONOMETRISCHE FUNKTIONEN

ATN — Gibt den arc. Tangens eines Wertes im Bogenmaß oder im Winkelmaß,
z.B.: $A = \text{ATN}(B)$.

COS — Gibt den Cosinus eines Wertes,
z.B.: $A = \text{COS}(B)$.

DEG — Alle folgenden trigonometrischen Funktionen werden im Winkelmaß verarbeitet,
z.B.: DEG.

RAD — Alle trigonometrischen Funktionen werden im Bogenmaß verarbeitet, sofern nicht DEG auszuführen ist.

SIN — Gibt den Sinus des Wertes,
z.B.: $A = \text{SIN}(B)$.

TAN — Gibt den Tangens eines Wertes.

BESONDERE FUNKTIONEN

ADR — Gibt die dezimale Speicheradresse des Anfangs einer Stringvariablen,
z.B.: $A = \text{ADR}(B\$)$.

ADR ("DIESES STRING"),
FRE — Gibt den (dezimalen) Wert des verfügbaren RAM's,
z.B.: $A = \text{FRE}(0)$ (Der Wert hat keinen Einfluß).
PRINT FRE(A).

PEEK — Gibt den Inhalt einer Speicherstelle,
z.B.: $A = \text{PEEK}(B)$.

USR — Ruft Programmiersprachen-Subroutinen von BASIC ab, **USR(ADDR, P1, P2, ..., PN)**, beispielsweise, schiebt die Argumente (P1 zu PN) in umgekehrter Reihenfolge auf den Stack. Somit wird das letzte Argument "PN" vor dem ersten Argument "P1" auf den Stack geschoben. Die Anzahl der Arguments — durch ein einziges Byte repräsentiert — wird daraufhin auf den Stack geschoben. Sind in der Funktion keine Argumente spezifiziert, wird Null auf den Stack geschoben.

Daraufhin wird die Programmiersprachen-Subroutine an der Adresse (ADDR) aufgerufen. Soll die Programmiersprachen-Routine zu einem Wert in BASIC zurückkehren, müssen die Low und High Bytes jeweils in den Speicherplätzen \$D4 und \$D5 gespeichert werden.

Die Routine muß die Argumentzahl und die Argumente vor dem Zurückkehren zu BASIC entfernen, oder es kommt zu einem System-Crash.

Beispiel: $A = \text{USR}(B, C, D)$ (Die Routine bei B wird aufgerufen, und die Parameter in C und D werden über den Stack zur Subroutine geleitet.)

STRING-FUNKTIONEN

ASC — Gibt den ASCII-Wert für das erste Zeichen eines Strings,
z.B.: $A = \text{ASC}("A")$ — (A ist 65)
 $A = \text{ASC}(B\$)$ (Stringvariablen sind auch gültig.)

CHR\$ — Gibt das durch den angesprochenen ASCII-Wert vertretene Zeichen. Reziproke Funktion von ASC,
z.B.: $A\$ = \text{CHR\$}(65)$ ($A\$ = "A"$)

LEN — Gibt die Länge der Stringvariablen,
z.B.: $A = \text{LEN}(A\$)$.

STR\$ — Gibt den String eines angesprochenen Wertes an. (formt eine Zahl in eine Kette um),
z.B.: $A\$ = \text{STR\$}(65)$ ($A\$ = "65"$).

VAL — Gibt den Wert einer Zahl n, die eine bestimmte Kette darstellt. (formt eine Kette in eine Zahl um),
z.B.: $A = \text{VAL}("100")$ ($A = 100$).

STRING-VERARBEITUNG

Im ATARI-BASIC werden keine String-Variablenfelder (String-Arrays) verarbeitet. Trotzdem sind Verknüpfungen ("Concatenation") möglich,
z.B.: Substrings

```
50 A$ = "WOLFGANGSTEPHENDAVID"
60 B$ = A$(9,16) (B$ ist "STEPHEN")
```

```
Verknüpfung (Concatenation)
50 A$ = "HALLO"
60 B$ = "FRED"
70 A$(LEN(A$) + 1) = B$ (A$ ist "HALLO FRED")
```

```
Suche nach einem String
50 FOR Z = 1 TO LEN(A$)
60 IF A$(Z,Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z
```

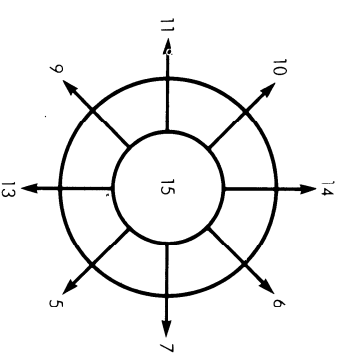
STEUERGERÄTE FUNKTIONEN

PADDDLE — Nennt die Position eines Paddles (Drehreglers). Die Paddles sind von vorn nach hinten von 0-3 nummeriert. Die erhaltene Zahl liegt je nach Stellung zwischen 1 (min.) und 228 (max.),
z.B.: $A = \text{PADDDLE}(0)$.

PTRIG — Gibt eine '0' bei gedrücktem oder eine '1' bei nicht gedrücktem Paddle,
z.B.: $A = \text{PTRIG}(0)$.

STICK — Nennt die Position eines Steuerknüppels (Joysticks). Die Steuerknüppel haben die Nummer(n) '0' und/oder '1', von vorne nach hinten

z.B.: $A = \text{STICK}(0)$ (siehe Zeichnung)
STRIG — Gibt eine '0' bei gedrücktem oder eine '1' bei nicht gedrücktem Steuerknüppel. Die Steuerknüppel haben die Nummer '0' und/oder '1',
z.B.: $A = \text{STRIG}(0)$.



SPEZIELLE FUNKTIONSTASTEN

ESC (ESCAPE) ist in der Funktion vom jeweiligen Programm abhängig.

BREAK dient zum Abbruch selbst-geschriebener Programme.

RESET unterbricht den Programm-Ablauf und führt zum Anfang zurück.

SET-CLR-TAB bewegt den Cursor zum nächsten, vorher festgelegten Tabulator-Stop.

SHIFT SET-CLR-TAB löscht eine Tabulator-Stelle.

CONTROL mit **1** unterbricht die Ausgabe auf den Bildschirm, noch-maliges Drücken setzt die Ausgabe fort.

CONTROL mit **2** läßt einen Summer ertönen.

CONTROL mit **3** markiert das Date-nende (EOF) nach der Eingabe.

BEARBEITUNGS- FUNKTIONEN

SHIFT mit **INSERT** fügt eine Leer-zeile oberhalb des Cursors ein.

CONTROL mit **INSERT** dient zum Einfügen von Leerzeichen.

DELETE BACK SPACE bewegt den Cursor nach links und löscht vor-handene Zeichen.

SHIFT mit **DELETE BACKSPACE** löscht die Zeile, in der sich der Cur-sor befindet.

CONTROL mit **DELETE BACK-SPACE** löscht das Zeichen unter dem Cursor. Die Zeichen rechts rücken nach.

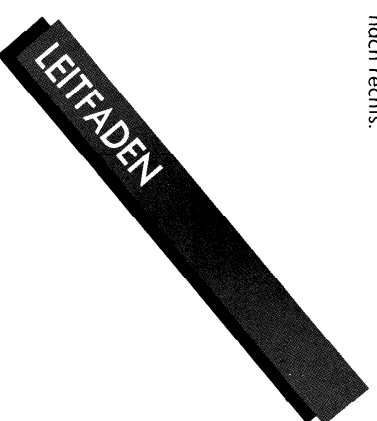
SHIFT mit **CLEAR** oder **CONTROL** mit **CLEAR** löscht den Bildschirm.

CONTROL **↑** bewegt den Cursor aufwärts.

CONTROL **↓** bewegt den Cursor abwärts.

CONTROL **←** bewegt den Cursor nach links.

CONTROL **→** bewegt den Cursor nach rechts.



FEHLERMELDUNGEN

Fehler- Nummer Bedeutung

- 2 Speicher voll
- 3 Zahlenbereich falsch
- 4 mehr als 128 Variablen
- 5 String zu lang
- 6 zu wenig Daten
- 7 Zahl größer als 32767
- 8 falscher Input-Befehl
- 9 DIM-Fehler
- 10 nicht ausführbar
- 11 Zahlenbereich verlassen
- 12 Zeile nicht gefunden
- 13 kein passender FOR-Befehl
- 14 Zeile zu lang
- 15 GOSUB/FOR entfernt
- 16 RETURN-Fehler/passen des GOSUB fehlt
- 17 Syntax-Fehler
- 18 ungültiges String-Zeichen
- 19 LOAD-Programm zu lang
- 20 Gerätenummer größer als 7
- 21 LOAD-Datei-Fehler

Anmerkung: Nachfolgend Eingabe/Ausgabe-Fehler, welche sich beim Anschluß von Diskettenstationen, Druckern oder anderen Zusatzgeräten ergeben können (weitere Angaben finden Sie bei der entsprechenden Hardware).

- 128 mit BREAK abgebrochen
- 129 IOCB schon geöffnet
- 130 Gerät nicht bekannt
- 131 IOCB „nur Ausgabe“ möglich
- 132 ungültiger "Händler"-Befehl
- 133 Gerät oder Datei nicht vorbereitet (OPEN fehlt)
- 134 ungültige IOCB-Nummer
- 135 IOCB-„nur Eingabe“ möglich
- 136 Datei-Ende mit EOF
- 137 Datensatz verstümmelt
- 138 Gerät antwortet nicht
- 139 Datenverkehr gestört
- 140 "Serial Bus"-Lesefehler
- 141 Cursor-Positionierung unzulässig
- 142 Datenaustausch gestört
- 143 Datenverkehr mit Fehler
- 143 der Prüfsumme

- 144 Diskette kann nicht gelesen/beschrieben werden
- 145 falscher Bildschirm-Modus
- 146 Funktion nicht vorgesehen
- 147 Speicher reicht nicht
- 160 Diskettenstation-Nr. falsch
- 161 zu viele Dateien geöffnet
- 162 Diskette voll
- 163 System-Fehler
- 164 Datei-Nummer paßt nicht
- 165 Datei-Namen-Fehler
- 166 POINT-Angaben falsch
- 167 Datei gesichert
- 168 falscher Geräte-Befehl
- 169 Disketten-Inhaltsverzeichnis voll
- 170 Datei nicht gefunden
- 171 POINT-Angaben ungültig
- 172 Diskette mit falschem DOS
- 173 Diskette kann nicht formatiert werden

©ATARI Inc. 1983. Alle Rechte vorbehalten