ATARI BASIC

REFERENCE GUIDE

LEITFADEN

MANUAL DE REFERENCIA

MANUALE D'USO

MANUEL DE REFERENCE

HANDLEIDING

ATARI®

ATARI BASIC

ATARI®

FOR EXPERIENCED PROGRAMMERS

Learning BASIC is like learning any other language—it takes a little time and effort, but the rewards are great. This guide provides information about ATARI BASIC—a popular, powerful dialect—for those who are already familiar with the BASIC programming language. This guide is intended for reference use only. It does not present comprehensive programming examples or tutorial information for the beginner. Both beginning and experienced programmers should refer to the following sources for more information: ATARI BASIC by Albrecht, Finkel, and Brown: ATARI BASIC REFERENCE MANUAL; and INSIDE ATARI BASIC by Bill Carris.

INDEX

OPERATOR PRECEDENCE

Operations within the innermost set of parentheses are performed first and proceed out to the next level. When sets of parentheses are enclosed in another set, they are said to be "nested". Operations on the same nesting level are performed in the following order:

HIGHEST PRECEDENCE TO LOWEST PRECEDENCE

<, >, =, <=, >=, <>
Relational operators used in string expressions have the same precedence and are performed from left to right.

-
Unary minus (denotes a negative number).

^
Exponentiation.

***, /**
Multiplication and division have the same precedence level and are performed from left to right.

+, -
Addition and subtraction have the same precedence level and are performed from left to right.

<, >, =, <=, >=, <>
Relational operations in numeric expressions have the same precedence level from left to right.

NOT
Unary operator

AND
Logical AND

OR
Logical OR

(Allowable abbreviation in parenthesis.)
The following words can be used as program statements, or as direct commands by typing them without a line number and pressing RETURN. These words may not be used as variable names.

SYSTEM CONTROL

BYE (B.) — Exits from BASIC to SELF TEST MODE

DOS — Displays DOS menu (use with a disk drive, only).

CSAVE (CS.) — Saves program file to Cassette.

CLOAD — Loads program file from Cassette

SAVE (S.) — Saves a BASIC program to an output device. Ex.: SAVE "D:MYFILE.BAS"

LOAD (LO.) — Loads a program file from an input device. Ex.: LOAD "D:MYFILE.BAS"

LIST (L.) — Sends a program list to screen or output device.
Ex.: LIST (lists whole program)
LIST 10 (lists line 10 on screen)
LIST 10, 20 (lists everything from line 10 to line 20)
LIST "P:" (list to printer)
LIST "P:", 10, 20 (lines 10-20 to printer)
LIST "D:MYFILE.LST" (list to a disk file)
LIST "D:MYFILE.LST",10,20 (list 10-20 to a disk file)
LIST "C:" (list to cassette)

ENTER (E.) — Enters a program from list input device
Ex.: ENTER "C:"
ENTER "D:MYFILE.LST"
Note: Common line numbers in a program that has already been loaded will be overwritten.

NEW — Clears program from memory.

RUN — Begins execution of a BASIC program. Program may be in memory or loaded from disk or tape. (Initializes variables to zero and undimensions arrays and strings.)
Ex.: RUN (executes program in memory)
RUN "D:MYFILE.BAS" (LOADs program from disk and executes it)

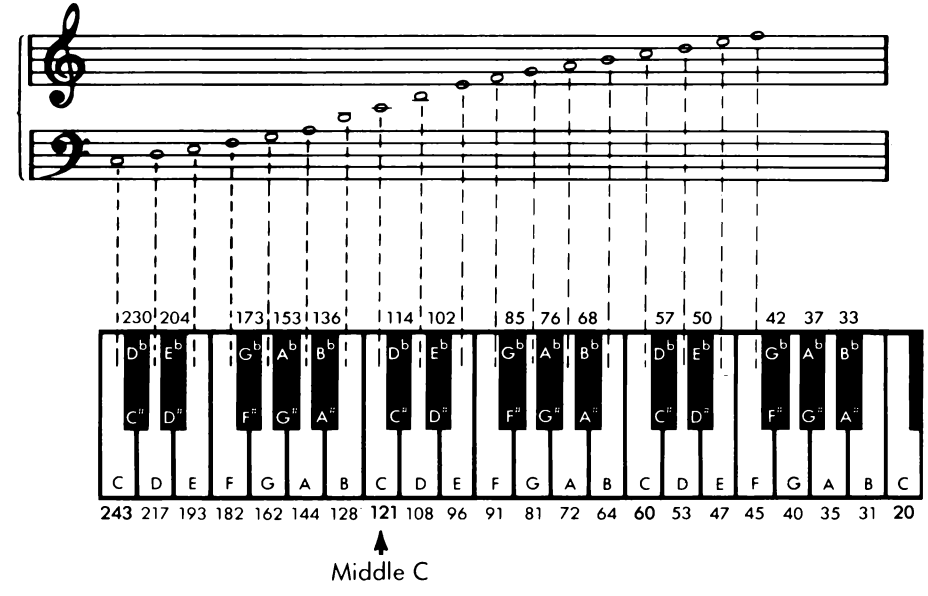
CONT — Continues program execution after the BREAK key has been pressed or the program has executed a STOP or END. If additional program statements are on the same line, they are not executed. The program continues execution with the next numbered line.

SOUND STATEMENT

SOUND (SO.) — Sets one of four channels to produce sound through the TV speakers. The sound continues until another SOUND statement addresses the same channel, or an END, RUN or NEW statement is executed. Channels are programmed independently and may all be on at once. This statement must be followed by four values (numbers, variables or expressions).

Ex.: SOUND A, B, C, D, where:
A = Channel number (0-3)
B = Period (0-255) The larger the value, the lower the frequency. Frequency=31960/(PERIOD + 1) See chart for musical equivalents.
C = Distortions (0-14, even #'s only) Ten and 14 are "pure" tones. Other numbers result in other noises.
D = Volumes (0-15) The larger the value, the louder the sound. 0 is off. If the total volume for all 4 voices exceeds 32, the speaker may "buzz".

RELATION OF PIANO KEYBOARD TO MUSICAL SCALE



PROGRAM CONTROL

GOTO (G.) — Program execution continues at the line number specified.
Ex.: GOTO 30 (execute program from line 30.)
Ex.: GOTO A+10 (Legal, but can be hard to debug.)

ON...GOTO — Program execution continues at the line number indicated by an expression.
Ex.: 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.) — Program execution continues at the line number specified. A RETURN statement will return control to the statement following the GOSUB.
Ex.: GOSUB 30 (execute subroutine at line 30)
GOSUB A + 10 (Legal, but can be very hard to debug.)

ON...GOSUB — Program execution continues at the line number indicated by an expression. A RETURN will return control to the statement following the ON...GOSUB.
Ex.: 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.)

Warning: If the expression evaluates a number less than 1 or greater than the number of line numbers, the results are unpredictable.

RETURN (RET.) — Ends a subroutine and returns control to the statement immediately following the last GOSUB statement executed.
Ex.: RETURN

FOR(F.) — This statement sets up the starting and ending values of an index variable and the value to be added to it each time a FOR...NEXT loop executes. The value added is 1 unless specified otherwise by a STEP statement. A NEXT statement will cause the instruction between FOR and NEXT to repeat.
Ex.: FOR A = 1 TO 10 (A will start at 1, increase by 1 and stop at 10.)
FOR A = 10 TO 1 STEP -2 (A negative STEP)
FOR A = B/T TO B*T STEP X (Computed values are valid, too.)

NEXT (N.) — Ends a FOR...NEXT loop. Checks that the index value has not passed the end value, adds the step value to the index value and continues execution at the statement after the FOR. If the index passes the end value, goes to the statement after the NEXT.
Ex.: NEXT A (A is the index variable.)

POP — Removes return information stored about the last FOR or GOSUB statement executed. Useful for leaving a FOR-NEXT loop early, or leaving a subroutine without executing the RETURN statement.
Ex.: POP: GOTO 10

IF...THEN — The statement after the THEN is executed when the condition between IF and THEN is true. Otherwise goes to the next program line.
Ex.: IF A ≠ C 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" (The expression "A is non-zero" is True.)

TRAP (T.) — On an error TRAP goes to the line specified. TRAP stays set until an error occurs or next trap statement. PEEK (195) returns the error number.
PEEK(187* 256 + PEEK(186) returns the line number.
Ex.: TRAP 30 (On an error, go to line 30.)
TRAP 40000 (Line numbers greater than 32767 turn TRAP off.)

STOP — Stops program. Prints line number. Does not close files or turn off sound. Program can be restarted with CONT.
Ex.: IF A = B THEN STOP

END — Stops program, closes all open files, turns off sound.

INPUT OUTPUT

(I/O) Device Names

Each device in the ATARI family has a unique device name. Disk drives and the ATARI 850 Interface Module (RS232 handler) require a device number (1-4). Disk drives also require a file name. File names must be enclosed in quotes or contained in string variables. Here are some examples:

K: Keyboard. Input only.
P: Printer. Output only.
C: Cassette. Input and Output.
S: Screen (TV). Output only.
E: Screen Editor (keyboard and screen combined). Input and Output.
R: RS232 Handler (ATARI 850 Interface Module). Input and Output.

D:FILENAME.EXT	Execute File "FILENAME.EXT" on disk drive #1.
D2:FILENAME.EXT	Same file on drive #2.

I/O STATEMENTS

Disk file names start with a letter and can be up to 8 characters long. The filename can end with an optional extension. (A period followed by 1-3 characters.) You can use any 3 letters or numbers you want. Some useful extensions are:

.BAS= Saves BASIC programs (same as SAVE).
.LST = Lists BASIC programs (same as LIST).
.DAT= data files
.OBJ= Machine language (object) file
.TXT = Text file

OPEN (O.) — Prepares a device for input or output. IOCB numbers are 1-7. Uses the following codes:

TYPE	CODE	OPERATION
Input	4	Read only.
Output	8	Write only.
Update	12	Read and write
Append	9	Add to end of file
Directory	6	Disk drive directory only

Ex.: OPEN #1,4,0,"K:" (Open keyboard for input on IOCB #1.)
OPEN #2,8,0 "P:" (Open printer for output on IOCB #2)
OPEN #1,12,0, "D:MYFILE.DAT" (Open disk file "MYFILE.DAT" for update on IOCB #1.)
OPEN #1,6,0, "D:*.*" (Open drive 1 for directory on IOCB #1.)

CLOSE (CL.) — Closes device after input or output operation and releases IOCB. May be executed when no device was opened. IOCB numbers are 1-7, as in OPEN.

Ex.: CLOSE #1 (Close file open on IOCB #1 and release the IOCB.)

INPUT (I.) — Gets a line of characters from device. Line must be terminated by a RETURN character.

Ex.: INPUT A (Get a number and put it in A)
INPUT A,B,C (Get 3 numbers, separated by commas, and put them in A, B & C.)
INPUT A$ (Get a string of characters and put it in A$, A$ will not contain a RETURN character.)
INPUT #1, A$, B (Get a character string from device open on IOCB #1 and put in A$ & B.)

PRINT (PR.) or **(?)** — Sends data to the screen or other device.

Ex.: PRINT (sends a blank line.)
PRINT "The number is"; A (Prints text and number to screen.)
PRINT "The number is", A (The comma causes A to be printed in a separate column. POKE 201 with the desired column width.
PRINT A$; (Semicolon prevents RETURN from being sent at end of line.)
PRINT #1, A$ (Send A$ to device open on IOCB #1.)

LPRINT (LP.) — Prints data on the printer. No open or close is necessary. Semicolon at end of line does not prevent RETURN from being sent.

Ex.: LPRINT (Send a blank line to printer.)
LPRINT A$ (A$ will be printed)
LPRINT A$;B (A$ and B will be on same line.)
LPRINT A$, B (The comma causes B to be printed in a separate column. POKE 201 with the desired column width.)

GET — Gets a single byte from device specified and puts it in the variable specified.

Ex.: Get #1,A (Gets a byte from device open on IOCB #1 and puts it in A).

PUT — Puts the single byte in the variable to the device specified.

Ex.: PUT #1,A (Puts the byte in A to the device open on IOCB #1.)

NOTE (NO.) — Used with disk to determine location of next byte to the read or written.

Ex.: NOTE #1, SEC, BYTE (Will put sector number in SEC and byte number in BYTE. Refers to file open on IOCB #1.)

POINT (P.) — Used to tell DOS the location where the next byte is to be read from or written to.

Ex.: POINT #1, SEC, BYTE (Will point DOS at sector number SEC, byte number BYTE.)

STATUS (ST.) — Gets status for specified device. Status code returned can be found in the error message list.

Ex.: STATUS #1, A (Put status of device open on IOCB #1 in A.)

PROCESSING STATEMENTS

LET — Assigns values to numeric or string variables.
Ex.: LET A=B (value of B assigned to A)
LET A$="HELLO"
A=B:A$="HELLO"(LET can be omitted.)

POKE — Puts a number between 0 and 255 into a specified memory location between 0 and 65535. PEEK is similarly used to read the memory location. Decimal values will
be rounded.
Ex.: POKE 82,0 (Puts 0 into memory location 82.)
A = PEEK(82) (Reads the contents of memory location 82 and puts it in A.)

DIM — Reserves space in memory for strings and numeric arrays. Each character space reserved for a string takes one byte; each element in a numeric array six bytes
Ex.: DIM A$(10) (a string variable with a length of 10 bytes.)
DIM B(10) (a numerical array; B contains elements 0-10)
DIM B(10,10) (a 2 dimensional array)
DIM A$(10),B(10) (Separate different variables by commas.)

COM — Same as DIM.

CLR — Clears the dimensions of any array, and clears strings, and numeric variables to zero (the opposite of DIM).
Ex.: CLR

DATA (D.) — Creates a list of numbers and/or letters to be used by the READ statement.
Ex.: DATA 1, 2, 3, 4, A, B, C, D (A list of information to be read.)

READ — Reads the next item in a DATA statement and assigns it to a variable. When one DATA statement has been used, READ will get data from the next DATA statement in the program.
Ex.: READ A (A will be the next number on the list in the DATA statement.)
READ A$ (Works for strings, too.)
READ A,A$,B,B$ (Separate multiple items by commas.)

RESTORE (RES.) — Points READ at a DATA statement.
Ex.: RESTORE (Next data item will be first item from first DATA statement.)
RESTORE 10 (Next data item will be first item from DATA statement in line 10.)

REM. (R.) or ([SPACE].), — Allows remarks. BASIC ignores everything from REM to the end of the line.
Ex.: REM This is a remark statement.

GRAPHICS

GRAPHICS (GR.) — Selects graphics mode, Mode + 16 selects a full screen (no text window). Mode + 32 does not clear screen. Modes 0-15 available.
Ex.: GRAPHICS 8 (Graphics mode 8 with text window.)
GRAPHICS 8 + 16 (Mode 8, full screen)
GRAPHICS 8 + 32 (Won't clear screen)
GRAPHICS 8 + 16 + 32 (both options combined)

SETCOLOR (SE.) — Sets the hue and luminance of the chosen color register. Register number is not the same as with COLOR command.
Ex.: SETCOLOR 1, 2, 4 (Set reg. 1 to hue 2 and luminance 4.)
Registers 0-4
Hues 0-15
Luminances 0-14, even numbers only (except GTIA modes, which can use both even and odd numbers up to 15).

COLOR (C.) — In map modes (3-11) selects a color register to use for PLOT. Register here is not the same as the register in SETCOLOR.
Ex.: COLOR 2 (Selects color register 2 in modes 0-2, selects ASCII character whose values is 2 for PLOT.)

TABLE OF MODES AND SCREEN FORMAT

SCREEN FORMAT

Graphics Mode	Mode Type	Columns	Rows — Split Screen	Rows — Full Screen	Number of Colors	RAM Required (Bytes) Split	Full
0	TEXT	40	—	24	1-1/2	—	992
1	TEXT	20	20	24	5	674	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

PLOT (PL.) — Puts a single point or character on the screen at a specified location.
Ex.: PLOT X, Y (X and Y must be positive coordinates.)

POSITION (POS.) — Selects a screen position, but nothing is plotted. Useful for positioning text with PRINT.
Ex.: POSITION X, Y

LOCATE (LOC.) — Retrieves data stored at a specified screen location. Gets characters in modes 0-2, color numbers in modes 3-11.
Ex.: LOCATE X, Y, D (Moves to X, Y and puts data in D.)

DRAWTO (DR.) — Draws a line between the last position of the cursor and the specified X & Y coordinates. The cursor ends up at the new coordinates.
Ex.: DRAWTO X, Y (draws a line to point X, Y from the current position of the cursor.

TABLE OF SETCOLOR "DEFAULT" COLORS*

Setcolor (Color Register)	Defaults To Color	Luminance	Actual Color
0	2	8	ORANGE
1	12	10	GREEN
2	9	4	DARK BLUE
3	4	6	PINK OR RED
4	0	0	BLACK

*"DEFAULT" occurs if no SETCOLOR statement is used.

Note: Colors may vary depending upon the television monitor type, condition, and adjustment.

THE ATARI HUE (SETCOLOR COMMAND) NUMBERS AND COLORS

Colors	Setcolor (aexp2) Numbers
GRAY	0
LIGHT ORANGE (GOLD)	1
ORANGE	2
RED-ORANGE	3
PINK	4
PURPLE	5
PURPLE-BLUE	6
BLUE	7
BLUE	8
LIGHT BLUE	9
TURQUOISE	10
GREEN-BLUE	11
GREEN	12
YELLOW-GREEN	13
ORANGE-GREEN	14
LIGHT ORANGE	15

Note: Colors vary with type and adjustment of TV or monitor used.

FUNCTIONS

A function takes one or more values and returns another value. Values may be strings or numbers. The examples show A (a numeric variable) or A$ (a string variable) as being equal to the function, but a function can be used almost anywhere you would use a value (including in another function).

ARITHMETIC FUNCTIONS

ABS — Returns absolute (unsigned) value of a number.
Ex.: A=ABS(B)

CLOG — Returns the logarithm to the base 10 (common log).
Ex.: A=CLOG(B)

EXP — Returns the value of "e" (approximately 2.718) raised to the power specified. In some cases, EXP is accurate only to 6 significant digits. This is the inverse of LOG.
Ex.: A = EXP(B)

INT — Returns the greatest integer less than or equal to a value.
Ex.: A = INT(B)

LOG — Returns the natural logarithm of a value. This is the inverse of EXP.

RND — Returns a random number between 0 and 1. Never returns a 1. Value makes no difference
Ex.: A = RND(0) (A = a number greater than or equal to 0 and less than 1.)
A = RND(0)*8 (A = a number greater than or equal to 0 and less than 8.)

SGN — Returns a -1 if value is negative, a zero if it's 0 and a 1 if it's positive.
Ex.: A = SGN(A)

SQR — Returns the positive square root of a positive value.
Ex.: A = SQR(B)

TRIGONOMETRIC FUNCTIONS

ATN — Returns the arctangent of a value in radians or degrees.
Ex.: A = ATN(B)

COS — Returns the cosine of a value.
Ex.: A = COS(B)

DEG — All subsequent trig functions will be in degrees.
Ex.: DEG

RAD — All subsequent trig functions will be in radians. The computer assumes you want radians unless you specify DEG (degrees).

SIN — Returns the sine of the value.
Ex.: A = SIN(B)

TAN — Gives the tangent of a value.

SPECIAL PURPOSE FUNCTIONS

ADR — Returns the decimal memory address of the beginning of a string.
Ex.: A = ADR(B$)
A = ADR("THIS STRING")

FRE — Returns the number of bytes of user RAM left. (0) is a required "dummy" variable.
Ex.: A =FRE(0)
PRINT FRE(0) (This will tell you how much memory is left.)

PEEK — Returns the number stored at a specific memory location. The address must be a number between 0 and 65535. The number returned will be between 0 and 255.
Ex.: A = PEEK(B)

USR — Calls machine language subroutines from BASIC. USR(ADDR, P1, P2,...,PN), for example, pushes the arguments (P1 to PN) onto the stack in reverse order. Thus, the last argument "PN" is pushed onto the stack before the first argument "P1".
The number of arguments — represented as a single byte — is then pushed to the stack. If there are no arguments specified in the function, zero is pushed to the stack. The machine language subroutine at address (ADDR) is then called. If the machine language routine is to return a value in BASIC, the Low and High bytes must be stored in memory locations $D4 and $D5, respectively.
The routine must remove the argument count and the arguments before returning to BASIC, or the system will crash.
Ex.: A = USR(B, C, D)
(The routine at B will be called, and the parameters in C and D will be passed to the subroutine via the stack.)

STRING FUNCTIONS

ASC — Returns the ATASCII code number for the first character of a string.
Ex.: A =ASC("A") (A will be 65.)
A = ASC(B$) (String variables can be used.)

CHR$ — Returns the character represented by the ATASCII code number specified. Reciprocal of ASC.
Ex.: A$ = CHR$(65) (A$ will be "A".)

LEN — Returns a number that represents the length of a string variable.
Ex.: A =LEN(A$)

STR$ — Returns a string representing a specified value. (Translates a number into a string.)
Ex.: A$ =STR$(65) (A$ will equal "65" which is not a number but a string.)

VAL — Returns a number representing a specific string. (Translates a string into a number.)
Ex.: A =VAL("100") (A will equal the number 100.)

STRING MANIPULATION

ATARI BASIC does not use a string array format for manipulating strings. A powerful mid-string command allows such things as concatenation of strings and other string handling.
Examples:
SUBSTRINGS
50 A$ = "DAVEMARKGRETCHEN"
60 B$ = A$(9,16) (B$ is "GRETCHEN")

CONCATENATION
50 A$ = "HI"
60 B$ = "FRED"
70 A$(LEN(A$)+1) = B$ (A$ is "HI FRED")

SEARCHING A STRING
50 FOR Z = 1 TO LEN(A$)
60 IF A$(Z,Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z

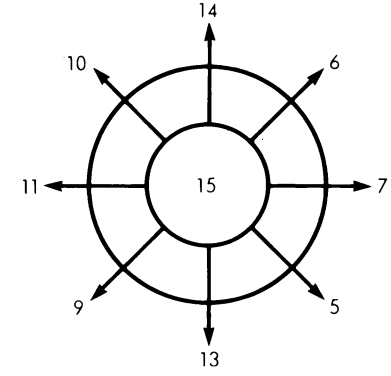
GAME CONTROLLER FUNCTIONS

PADDLE — Returns the position of a specific paddle controller. The paddles are numbered 0-3 from front to back. Number returned is between 1 and 228, increasing as the knob is turned left (counterclockwise).
Ex.: A = PADDLE (0)

PTRIG — Returns a 0 if a specific paddle trigger is pressed and a 1 if it isn't. The paddles are numbered 0-3 from front to back.
Ex.: A =PTRIG(0)

STICK — Returns the status of a specific joystick controller. The joysticks are numbered 0-1 from front to back. See diagram for details.
Ex.: A =STICK(0)

STRIG — Returns a 0 if a specific joystick trigger button is pressed and a 1 if it isn't. The joysticks are numbered 0 and 1.
Ex.: A = STRIG(0)



SPECIAL FUNCTION KEYS

ESC Causes next key pressed to be displayed as an international character or a graphics character, and executed when it's printed to the screen.

BREAK Causes a BASIC program to stop.

RESET Stops a running program, returns the screen to graphics mode 0, clears the screen, and terminates your files without closing them. It does not erase your program.

SET-CLR-TAB Moves the cursor to the next preset tab stop.

SHIFT SET-CLR-TAB Clears a tab.

CTRL SET-CLR-TAB Sets a tab.

CTRL 1 Stops and starts screen display scrolling.

CTRL 2 Sounds buzzer.

CTRL 3 Indicates end of file.

EDITING

SHIFT INSERT Inserts space for a line.

CTRL INSERT Inserts space for a character.

DELETE BACK S Deletes character to the left of cursor and moves cursor into the empty position.

SHIFT DELETE BACK S Deletes a line.

CTRL DELETE BACK S Deletes character at the cursor and moves remainder of line to fill in the empty position.

SHIFT CLEAR or **CTRL CLEAR** Clears screen.

CTRL UP ARROW Moves cursor up.

CTRL DOWN ARROW Moves cursor down.

CTRL LEFT ARROW Moves cursor left.

CTRL RIGHT ARROW Moves cursor right.

ERROR CODE

ERROR CODE	ERROR CODE MESSAGE
2	Memory Insufficient
3	Value Error
4	Too Many Variables
5	String Length Error
6	Out of Data Error
7	Number greater than 32767
8	Input Statement Error
9	Array or String DIM Error
10	Argument Stack Overflow
11	Floating Point Overflow/Underflow Error
12	Line Not Found
13	No Matching FOR Statement
14	Line Too Long Error
15	GOSUB or FOR Line Deleted
16	RETURN Error
17	Syntax Error
18	Invalid String Character
19	LOAD program Too Long
20	Device Number Larger
21	LOAD File Error

Note: The following are INPUT/OUTPUT errors that result during the use of disk drives, printers, or other accessory devices. Further information is provided with the auxillary hardware.

128	BREAK Abort
129	IOCB
130	Nonexistent Device
131	IOCB Write Only
132	Invalid Handler Command
133	Device or File not Open
134	BAD IOCB Number
135	IOCB Read Only Error
136	EOF
137	Truncated Record
138	Device Timeout
139	Device NAK
140	Serial Bus
141	Cursor Out of Range
142	Serial Bus Data Frame Overrun
143	Serial Bus Data Frame Checksum Error
144	Device Done Error
145	Bad Screen Mode Error
146	Function Not Implemented
147	Insufficient Screen RAM
160	Drive Number Error
161	Too many OPEN Files
162	Disk Full
163	Unrecoverable System Data I/O Error
164	File Number Mismatch
165	File Name Error
166	POINT Data Length Error
167	File Locked
168	Invalid Device Command
169	Directory Full
170	File Not Found
171	POINT Invalid
172	Illegal Append
173	Bad Format

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

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.)— Geht 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 Tastaturformat 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 eingeladen 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)

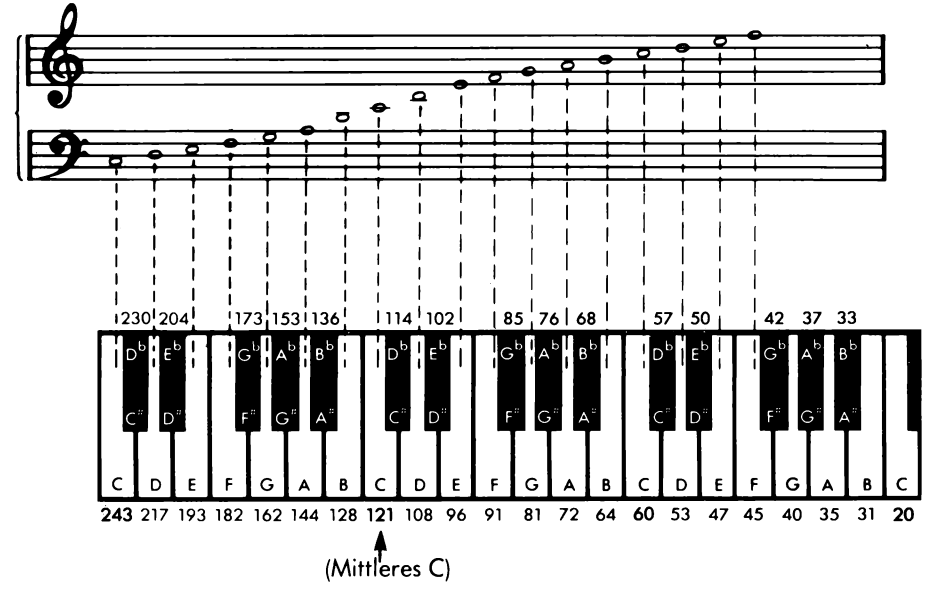
CONT — Führt ein Programm mit der nächsten numerischen Zeile weiter, nachdem die BREAK-Taste gedrückt wurde oder das Programm bis zum STOP oder END ausgeführt worden ist.

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: Wertet 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 Index-Wert 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 Tonkäle 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 (ATARI 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)

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 wahlweisen Erweiterung enden (ein Punkt, gefolgt von 1 - 3 beliebigen Buchstaben/Ziffern). Einige praktische Erweiterungen sind:

.BAS = Mit der Funktion "SAVE" versehenes BASIC-Progr.
.LST = Mit der Funktion "LIST" versehenes BASIC-Progr.
.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 IOCD #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 Kommata zu trennende 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 vorbereitetem 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 auszuschreibenden 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 auszuschreibende 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-Codei 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 Speicherstelle 82 ein)
A = PEEK (82) (Liest den Inhalt von Speicherstelle 82 und setzt ihn in Variable 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-Variable mit einer Länge von 10 Bytes)
DIM B (10) (Ein numerisches Variablenfeld; B umfaßt die Elemente 0 bis 10)
DIM B (10, 10) (ein 2-dimensionales Variablenfeld)
DIM A$ (10), B (10) (verschiedene Variablenfelder sind durch Kommata 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 verwendenden 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 Kommata 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 Anmerkungen. 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 Textfenster); Modus +32 löscht den Bildschirm nicht.
z.B.: GRAPHICS 8 (Graphik-Modus 8 mit Textfenster)
GRAPHICS 8 +16 (Modus 8, ungeteilter 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 Farbregisters. Die Registernummer ist nicht dieselbe 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 Verwendung 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 Blidschirm	Zeilen/ Ungeteilter Bildschirm	Anzahl Farben	Erforderliches RAM (Bytes) geteilt	ungeteilt
0	TEXT	40	—	24	1-1/2		992
1	TEXT	20	20	24	5	674	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

PLOT (PL.) — Setzt einen einzelnen Punkt oder ein Zeichen an eine bestimmte Stelle auf dem Bildschirm,
z.B.: PLOT X, Y (X-Y Koordinaten, nur positive Werte)

POSITION (POS.) — Wählt eine Bildschirmposition, es wird jedoch nicht aufgezeichnet. 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 Bildschirm 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 zwischen der letzten Position des Cursors (Markierungszeichen) und der spezifizierten 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).

TABELLE DER SETCOLOR "DEFAULT"(FEHL-) FARBEN

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

DIE ATARI-FARBEN (PAL)

Farben	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

FUNKTIONEN

Eine Funktion ist eine veränderliche Größe, die in ihrem Wert von einer anderen abhängig ist. Werte können Strings, Zahlen oder mathematische Ausdrücke sein. Die Beispiele zeigen A (eine numerische Variable) oder A$ (eine Stringvariable), 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 spezifizierten 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).

TRIGONOMETRISCHE FUNKTIONEN

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

COS — Gibt den Cosinus eines Wertes,
z.B.: A = 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 = SIN (B).

TAN — Gibt den Tangens eines Wertes.

BESONDERE FUNKTIONEN

ADR — Gibt die dezimale Speicheradresse des Anfangs einer Stringvariablen,
z.B.: A = ADR (B$).
A = ADR ("DIESES STRING").

FRE — Gibt den (dezimalen) Wert des verfügbaren RAM's,
z.B.: A = FRE (0). (Der Wert hat keinen Einfluß).
PRINT FRE (A).

PEEK — Gibt den Inhalt einer Speicherstelle,
z.B.: A = 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 and $D5 gespeichert werden.

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

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

STRING-FUNKTIONEN

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

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

LEN — Gibt die Länge der String variablen,
z.B.: A = LEN (A$).

STR$ — Gibt den String eines angesprochenen Wertes an. (formt eine Zahl in eine Kette um).
z.B.: A$ = 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 = 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

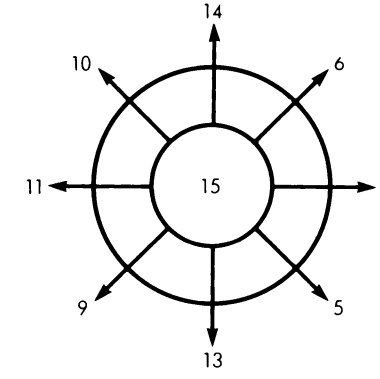
STEUERGERÄTE FUNKTIONEN

PADDLE — Nennt die Position eines Paddles (Drehreglers). Die Paddles sind von vorn nach hinten von 0–3 numeriert. Die erhaltene Zahl liegt je nach Stellung zwischen 1 (min.) und 228 (max.),
z.B.: A = PADDLE (0).

PTRIG — Gibt eine '0' bei gedrücktem oder eine '1' bei nicht gedrücktem Paddle,
z.B.: A = 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 = 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 = STRIG (0).



SPEZIELLE FUNKTIONSTASTEN

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

BREAK dient zum Abbruch selbstgeschriebener 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, nochmaliges Drücken setzt die Ausgabe fort.

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

CONTROL mit **3** markiert das Datenende (EOF) nach der Eingabe.

BEARBEITUNGS-FUNKTIONEN

SHIFT mit **INSERT** fügt eine Leerzeile oberhalb des Cursors ein.

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

DELETE BACK SPACE bewegt den Cursor nach links und löscht vorhandene Zeichen.

SHIFT mit **DELETE BACKSPACE** löscht die Zeile, in der sich der Cursor 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 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	zuviele Dateien geöffnet
162	Diskette voll
163	System-Fehler
164	Datei-Nummer paßt nicht
165	Datei-Namen-Fehler
166	POINT-Angaben falsch
167	Datei gesichtert
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

PARA PROGRAMADORES EXPERIMENTADOS

Aprender el BASIC es igual que aprender cualquier otro lenguaje; hay que dedicar algún tiempo, lo que exige cierto esfuerzo, pero el resultado vale la pena. Este manual facilita información accerca del ATARI BASIC, un dialecto apreciado y poderoso, destinado para aquellos que están acostumbrados ya al lenguaje de programación BASIC.
Este manual es sólo una guía de referencia. No presenta una serie completa de ejemplos de programación ni constituye un manual completo para uso de los principiantes.
Los programadores debutantes, así como los experimentados deberían consultar las obras indicadas a continuación para obtener más información:
ATARI BASIC, por Alrecht, Finkel & Brown: MANUAL DE REFERENCIA ACERCA DEL ATARI BASIC "INSIDE ATARI BASIC", por Bill Carris.

INDICE

ORDEN DE PRIORIDAD EN LA OPERACION

Las operaciones que se hallan entre paréntesis y están situadas más hacia el interior deben ser ejecutadas en primer lugar y conducen al siguiente nivel. Cuando un signo de paréntesis está incluido dentro de otro signo de paréntesis, los primeros se denominan "encajados". Las operaciones que están en el mismo nivel de encajamiento son ejecutadas en el siguiente orden:

MAYOR AL MENOR PRECEDENCIA

<, >, =, <=, >=, <>
Operadores relacionales utilizados en las expresiones en cadena. Tienen el mismo orden de precedencia y son ejecutados de izquierda a derecha.

-
Signo unario negativo.

^
Elevación a potencias

***, /**
Multiplicación y división tienen el mismo orden de precedencia y se ejecutan de izquierda a derecha.

+, -
Adición y sustracción tienen el mismo orden de precedencia y se ejecutan de izquierda a derecha.

En las expresiones numéricas, las operaciones relacionales tienen el mismo orden de precedencia de izquierda a derecha.

NOT
Operador unario

AND
AND lógico

OR
OR lógico

PALABRAS RESERVADAS

(Las Abreviaciones permitidas están entre paréntesis)
Los ordenadores ATARI no hacen una distinción entre las órdenes y las instrucciones. Las palabras indicadas a continuación pueden ser utilizadas como instrucciones de programa, o bien directamente, escribiéndolas y pulsando la tecla RETURN. Estas palabras no pueden ser empleadas como nombres de variables.

MANDO DEL SISTEMA

BYE (B.) — Sale del modo BASIC para pasar al modo SELF TEST.

DOS — Presenta el menú DOS (utilice sólo con una unidad de discos).

CSAVE (CS.) — Conserva el programa en cassette.

CLOAD — Carga el programa a partir de una cassette.

SAVE (S.) — Conserva un programa en BASIC, en un accesorio de salida.
Ej.: SAVE "D: MYFILE.BAS."

LOAD (LO.) — Carga un programa desde un accesorio de entrada.

LIST (L.) — Inscribe un programa en la pantalla o en un accesorio de salida.
Ej.: LIST (inscribe la totalidad del programa).
LIST 10 (inscribe la línea 10 en la pantalla).
LIST 10, 20 (inscribe todo lo que se encuentra entre las líneas 10 y 20 ambas inclusive).
LIST "P" (inscripción en el impresor).
LIST "P", 10, 20 (las líneas 10-20 se inscriben en el impresor).
LIST "D:MYFILE.LST", 10, 20 (inscripción de las lineas 10 a 20 en un archivo de disco).
LIST "C": (inscripción de todo el programa en cassette).

ENTER (E.) — Introduce el programa desde el accesorio de entrada.
Ej.: ENTER "C:"
ENTER "D:MYFILE.LST"
Las líneas que tienen el mismo número en un programa que ya ha sido cargado serán sobreescritas.

NEW — Borra el programa que está en la memoria.

RUN — Pone en marcha la ejecución de un programa BASIC. El programa puede estar en memoria o bien ser cargado a partir de un disco o cinta. (Marca las variables hasta cero, y reduce las tablas y cadenas).
Ej.: RUN (ejecuta el programa en memoria).
RUN "D:MYFILE.BAS" (Carga el programa a partir del disco y lo ejecuta).

CONT — Reanuda la ejecución del programa después de que haya sido pulsada la tecla de interrupción o después que el programa haya ejecutado un STOP o un END. Si existen otras instrucciones de programa en la misma línea, éstas no son ejecutadas. La ejecución del programa continúa en la siguiente línea numerada.

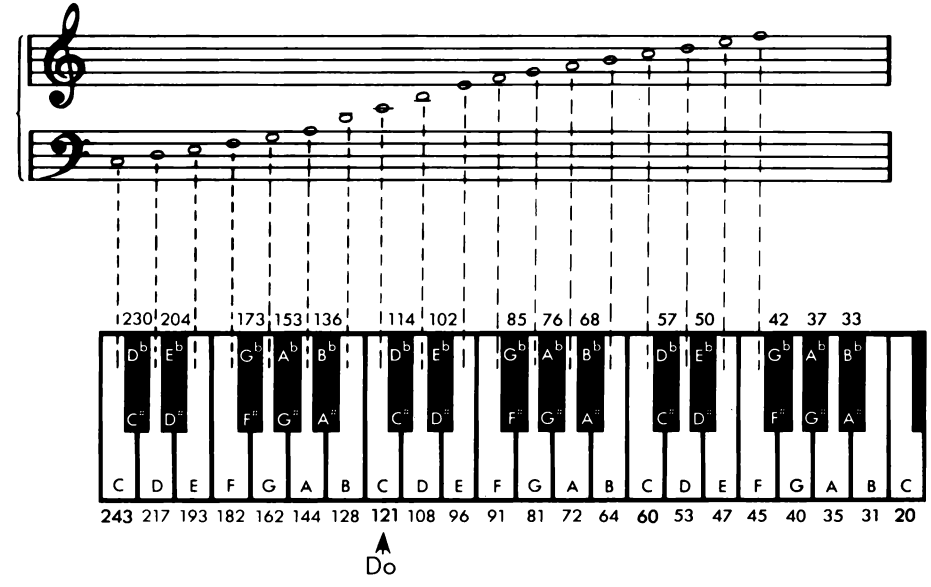
INSTRUCCIONES SONORAS

SOUND (SO.) — Produce un sonido en el altavoz del televisor por uno de los cuatro canales. El sonido continúa hasta que es dirigida otra instrucción de SOUND al mismo canal, o hasta que sea ejecutada una instrucción END, RUN o NEW. Los canales son programados independientemente y pueden funcionar todos simultáneamente. Esta instrucción debe ser seguida por cuatro valores (números, variables o expresiones).

Ej.: SOUND A, B, C, D, donde:
A = Número de canal (0 a 3)
B = Período (0 a 255). Cuanto mayor es el período, más baja es la frecuencia. Frecuencia = 31960/(PERIODO + 1). Véa la tabla para las equivalencias musicales.
C = Distorsión (0 a 14, números pares sólamente). 10 y 4 son tonos "puros". Los otros números producen otros ruidos.
D = Volúmen (0 a 15). Cuanto mayor es el número, más alto es el volúmen. 0 es desconección. Es posible que el altavoz "zumbe" si el volúmen total de las 4 voces es superior a 32.

(Ilustración de una pauta y un teclado)

RELACION DEL TECLADO DEL PIANO A LA ESCALA MUSICAL



MANDO DEL PROGRAMA

GOTO (G.) — La ejecución del programa continúa en la línea cuyo número ha sido especificado.
Ej.: GOTO 30 (ejecute el programa desde la línea 30).
Ej.: GOTO A + 10 (autorizado, pero puede ser difícil comprobar las instrucciones del programador).

ON GOTO — La ejecución del programa continúa en el número de la línea indicada por una expresión.
Ej.: 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.) — La ejecución del programa continúa en la línea especificada. Una instrucción de RETURN hace que el mando vuelva a la instrucción que sigue al GOSUB.
Ej.: GOSUB 30 (ejecuta un subprograma en la línea 30).
GOSUB A + 10 (autorizado, pero puede ser muy difícil comprobar las instrucciones del programador).

ON. . .GOSUB — La ejecución del programa continúa en la línea indicada por una expresión. Un RETURN hace volver el mando a la instrucción que sigue al ON . . . GOSUB.
Ej.: 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).
Aviso: Si la expresión evalúa un número menor de 1 o mayor de la cantidad de números de línea, los resultados son imprevisibles.

RETURN (RET.) — Termina un subprograma y hace volver el mando a la instrucción que sigue inmediatamente a la última instrucción GOSUB ejecutada.
Ej.: RETURN

FOR (F.) — Esta instrucción fija los valores inicial y final de una variable de índice, así como el valor que debe serle añadido cada vez que un FOR. . .NEXT ejecuta un bucle. El valor añadido es 1 al menos que sea especificado de otro modo mediante una instrucción STEP. Una instrucción NEXT hace que sean repetidas las instrucciones entre FOR y NEXT.
Ej.: FOR A=1 TO 10 (A comenzará en 1 y aumentará 1 cada vez hasta llegar a 10)
FOR A=10 TO 1 STEP-2 (Que es un STEP negativo)
FOR A=B/T TO B*T STEP X (Los valores calculados son también válidos)

NEXT (N.) — Termina un bucle FOR . . . NEXT. Verifica que el valor de índice no haya superado el valor final, incrementa el valor del índice por el valor de STEP y continúa la ejecución de una instrucción después del FOR. Si el índice sobrepasa el valor final, pasa a la insrucción después del NEXT.
Ej.: NEXT A (A es la variable del índice).

POP — Retira la información de regreso almacenada que concierne a la última instrucción FOR o GOSUB ejecutada. Es práctico para salir prematuramente de un bucle FOR—NEXT o para salir de un subprograma sin ejecutar RETURN.
Ej.: POP: GOTO 10

IF. . .THEN — La instrucción que sigue al THEN es ejecutada cuando la condición que está entre el IF y el THEN es verdadera. De otro modo, pasa a la siguiente línea del programa.
Ej.: IF A ≠ C THEN GOTO 300
IF A = B THEN PRINT
"A = B": PRINT "QUE LE PARECE!"
LET A = 5: GOTO 20
IF A THEN PRINT "A ES DISTINTO DE CERO" (Una expresión distinta de cero es verdadera).

TRAP (T.) — Al producirse un error, TRAP va a la línea especificada. TRAP permanece conectado hasta que se presente un error o bien hasta la próxima instrucción TRAP. PEEK (195) devuelve el número de error.
PEEK (187* 256 + PEEK(186) devuelve el número de línea)
Ej.: TRAP 30 (en caso de error, vaya a la línea 30).
TRAP 40.000 (los números de líneas superiores a 32.767 desconectan a TRAP).

STOP — Interrumpe el programa. Imprime el número de línea. No cierra los archivos, ni desconecta el sonido. Se puede reiniciar un programa con CONT.
Ej.: If A = B THEN STOP

END — Interrumpe el programa, cierra todos los archivos abiertos, desconecta el sonido.

El programa puede volver a ser puesto en marcha con CONT.

ENTRADA Y SALIDA (I/O) DESIGNACION DE APARATOS

Cada aparato de la familia ATARI tiene su propio nombre. Las unidades de discos y el módulo de interface ATARI 850™ Interface Module (manipulador RS232) deben tener un número de aparato (1-4). Las unidades de disco deben tener también un nombre de aparato. Los nombres de los aparatos deben estar incluídos entre comillas comprendidos en las variables de cadenas. Hay aquí algunos ejemplos:

K: Teclado, Entrada solamente.
P: Impresor. Salida solamente.
C: Cassette. Entrada y salida.
S: Pantalla (TV). Salida solamente.
E: Editor de pantalla (teclado y pantalla combinados). Entrada y Salida.
R: Manipulador RS232 (Módulo de interface ATARI 850).
D:FILENAME.EXT Esigue archivo "FILENAME.EXT su disk drive #1"
D2:FILENAME.EXT El mismo archivo en la unidad #2.

Los nombres de los archivos en disco comienzan con una letra y pueden tener hasta 8 caracteres. El nombre del archivo puede terminar con una extensión opcional. (Un punto seguido de 1 a 3 caracteres). Puede usted utilizar cualquier combinación de 3 letras o cifras según desee. Hay aquí algunas expresiones útiles:
.BAS = Salvaguarda los programas en BASIC (al igual que SAVE)
.LST = Lista los programas en BASIC (al igual que LIST)
.DAT = Archivos de datos
.OBJ = Archivo del lenguaje de la máquina (objeto)
.TXT = Archivo de texto.

INSTRUCCIONES DE ENTRADA/SALIDA I/O

OPEN (O.) — Prepra un accesorio para la entrada o salida. Los números del IOCB son 1 a 7. Utiliza los siguientes códigos:

TIPO	CODIGO	OPERACION
Entrada	4	Leer solamente
Salida	8	Escribir solamente
Actualización	12	Leer y escribir
Apéndice	9	Añadir al final del archivo
Directorio	6	Directorio de la unidad de disco solamente.

Ej.: OPEN # 1, 4, 0, "K": (Abrir el teclado para entrada en IOCB #1)
OPEN # 2, 8, 0, "P": (Abrir el impresor para entrada en IOCB #2)
OPEN # 1, 12, 0, "D: MYFILE.DAT" (Abrir el archivo "MYFILE.DAT" de disco para actualización en IOCB #1)
OPEN # 1,6,0, "D:*.*" (Abrir la unidad de disco # 1 para el directorio en IOCB #1)

CLOSE (CL.) — Cierra un accesorio después de la operación de entrada o salida y libera el IOCB. Puede ser ejecutado cuando no ha sido abierto ningún accesorio. Los números del IOCB son del 1 al 7, como en la operación OPEN.
Ej.: CLOSE # 1 (Cerrar el archivo abierto en IOCB # 1 y liberar el IOCB).

INPUT (I.) — Recibe una línea de caracteres de un accesorio. La línea debe acabar con un carácter RETURN.
Ej.: INPUT A (Buscar un número y colocarlo en A).
INPUT A,B,C (Buscar 3 números separados por comas y colocarlos en A, B y C).
INPUT A$ (Buscar una línea de caracteres y colocarla en A$; A$ no contiene el carácter RETURN).
INPUT #1, A$ 1B (Buscar una línea en el accesorio abierto en IOCB # 1 y colocarla en A$ y B).

PRINT (PR) o **(?)** — Envía datos a la pantalla o a otros accesorios.
Ej.: PRINT (envía una línea en blanco)
PRINT "El número es"; A (presenta el texto y el número en la pantalla)
PRINT "El número es", A (La coma hace que A se imprima en unâ columna separada POKE 201 con el ancho deseado de columna.)
PRINT A$; (el punto y coma impide que RETURN sea enviado al final de la línea).
PRINT #1;A$ (envía A$ al dispositivo abierto en IOCB #1)

LPRINT (LP) — Imprime los datos en el impresor. No es necesario abrir ni cerrar. El punto y coma al final de la línea no impide el envío de RETURN.
Ej.: LPRINT (envia linea blanca al impresor)
LPRINT A$;B (A$ y B) estarán en la misma linea.
LPRINT A$;B (La coma hace que B se imprima en una columna separada. POKE 201 con el ancho deseado de columna)

GET — Obtiene un solo byte del accessorio especificado y lo coloca en la variable especificada.
Ej.: GET #1,A (obtiene un byte del accesorio abierto en IOCB #1 y lo coloca en A)

PUT — Pone el byte en la variable del accesorio especificado.
Ej.: PUT #1,A (pone el byte en A en el accesorio abierto en IOCB #1)

NOTE (NO.) — Se utilizado con el disco para determinar la posición del siguiente byte que debe ser leído o escrito.
Ej.: NOTE #1, SEC, BYTE (Colocará el número de sector en SEC y el número de byte en BYTE. Se refiere al archivo abierto en IOCB #1)

POINT (P.) — Se utiliza para indicar a DOS la posición en que el siguiente byte debe ser leído o escrito.
Ej.: POINT #1, SEC, BYTE (dirigirá dos al número de sector SEC y al número de BYTE)

STATUS (ST.) — Obtiene el estado para el accesorio especificado. El código de estado se encuentra en la lista de mensajes de error.
Ej.: STATUS #1,A (Pone el estado del accesorio abierto en IOCB #1 en A)

INSTRUCCIONES DE PROCESAMIENTO

LET — Asigna valores a las variables numéricas o de cadena.
Ej.: LET A=B (valor de B asignado a A)
LET A$="HELLO"
A=B: A$="HELLO" (LET puede ser omitido)

POKE — Coloca un número entre 0 y 255 en una posición de memoria especificada entre 0 y 65.535. Los valores decimales son redondeados. Use PEEK para leer la memoria.
Ej.: POKE 82, 0 (Coloca 0 en la posición de memoria 82).
A= PEEK 82 (leer el contenido de la posición de memoria 82 y lo ponga en A).

DIM — Reserva espacio de memoria para las cadenas y tablas numéricas. Cada espacio de carácter reservado para una cadena toma un byte y cada elemento de una tabla toma seis.
Ej.: DIM A$ (10) (una variable de cadena)
DIM B (10) (una cadena numérica; B contiene los elementos 0 a 10)
DIM B (10, 10) (una tabla en 2 dimensiones)
DIM A$ (10), B (10) (separa los diferentes elementos por medio de comas)

COM — Como en DIM.

CLR — Cancela las dimensiones de cualquier ordenación, cancela las series de datos, y reduce las variables numéricas a cero.
Ej.: CLR

DATA (D.) — Crea una lista de números y/o letras que será utlilzada más adelante por la instrucción READ
Ej.: DATA 1,2,3,4,A,B,C,D (una lista de información para leer).

READ — Lee el próximo elemento en una instrucción DATA y lo asigna a una variable. Cuando está utilizado una instrucción DATA, READ toma los datos de la próxima instrucción DATA en el programa.
Ej.: READ A (A será el número siguiente en la lista de las instrucciones DATA).
READ A$ (válido también para las cadenas).
READ A, A$, B, B$ (Separa los elementos múltiples mediante comas).

RESTORE (RES.) — Dirige READ hacia una instrucción DATA.
Ej.: RESTORE (El siguiente dato será el primer elemento de la primera declaración de DATA).
RESTORE 10 (el proximo dato será el primer elemento de la instrucción DATA en la línea 10).

REM (R.) o ([ESPACIO].) — Permite hacer observaciones. El BASIC ignora todo lo que recibe de REM hasta el final de la línea.
Ej.: REM. Es una instrucción que formula una observación.

GRAFICOS

GRAPHICS (GR) — Selecciona el modo gráfico. Modo + 16 selecciona la pantalla completa (sin ventana de texto), mientras que Modo + 32 no borra la pantalla. Los Modos 0 a 15 son para el 1200XL.
Ej.: GRAPHICS 8 (modo gráfico 8 con ventana de texto).
GRAPHICS 8 + 16 (modo 8, pantalla completa).
GRAPHICS 8 + 32 (no borra la pantalla).
GRAPHICS 8 + 16 + 32 (combina las dos opciones).

SETCOLOR (SE) — Define el matiz y la luminosidad en el registro de colores elegido. El número de registro no es el mismo que con el mando COLOR.
Ej.: SETCOLOR 1,2,4 (Fija el registro 1 en el matiz 2 y la luminosidad 4)
Registros 0 a 4
Matices 0 a 15
Luminosidad 0 a 14, números pares sólamente (excepto los modos GTIA, que pueden utilizar números pares e impares hasta 15).

COLOR (C) — En los modos 3 a 11 para mapas, selecciona un registro de colores para emplearlo en PLOT. En este caso, el registro no es el mismo que en SETCOLOR.
Ej.: COLOR 2 (Selecciona el registro de colores 2 en modos 0 a 2, selecciona el carácter ASCII cuyo valor es 2 para PLOT).

CUADRO DE MODOS Y FORMATOS DE PANTALLA

FORMATO DE PANTALLA

Modo de Gráficas	Tipo de Modo	Columnas	Filas-Pantalla Dividida	Filas-Pantalla Entera	Número de Colores	RAM Requerida (Bytes) Dividida	Entera
0	TEXT	40	—	24	1-1/2	—	992
1	TEXT	20	20	24	5	674	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

PLOT (PL) — Coloca un solo punto o carácter en la pantalla, en un lugar especificado.
Ej.: PLOT X,Y (X e Y son las coordenadas que tienen que ser valores positivos).

POSITION (POS) — Selecciona una posición en la pantalla, sin trazar nada. Es útil para la colocación del texto con PRINT.
Ej.: POSITION X, Y.

LOCATE (LOC) — Extrae los datos almacenados en un lugar específico de la pantalla. Obtiene los caracteres en modos 0 a 2 y los números de colores en modos 3 a 11.
Ej.: LOCATE X,Y,D (Desplaza X,Y y coloca los datos en D).

DRAWTO (DR) — Traza una línea entre la última posición del cursor y el punto en las coordenadas X e Y especificadas. El cursor llegará a estas nuevas coordenadas y permanecerá en ellas.
Ej.: DRAWTO X.Y (traza una línea desde la posición presente del cursor hasta el punto en que se encuentran las coordenadas X e Y).

CUADRO DE "DEFECTOS" DE COLORES DE SETCOLOR

Setcolor (Registro de color)	Cambia al Color	Luminosidad	Color Real
0	2	8	NARANJA
1	12	10	VERDE
2	9	4	AZUL OSCURO
3	4	6	ROSA O ROJO
4	0	0	NEGRO

* El "Defecto" occurre en caso de no utilizarse ninguna instrucción de SETCOLOR.

Nota: Los colores pueden variar según el tipo de receptor de TV, su estado y ajuste.

THE ATARI HUE (SETCOLOR COMAND) NUMBERS AND COLORS

Colores	Numeros de Setcolor (a exp 2)
GRIS	0
NARANJA CLARO (DORADO)	1
NARANJA	2
ROJO ANARANJADO	3
ROSA	4
MORADO	5
MORADO AZULADO	6
AZUL	7
AZUL	8
AZUL CLARO	9
TURQUESA	10
VERDE AZULADO	11
VERDE	12
AMARILLO VERDOSO	13
NARANJA VERDOSO	14
NARANJA CLARO	15

FUNCIONES

Una función toma uno o varios valores y da como resultado otro valor. Los valores pueden ser cadenas o números (cifras). En los ejemplos, A (variable numérica) o A$ (variable de cadena) son iguales a la función, pero una función puede ser utilizada en casi todos los casos en que utilizaría usted un valor, incluso en otra función.

FUNCIONES ARITMÉTICAS

ABS — Da el valor absoluto (sin signo) de un número.
Ej.: A = ABS (B)

CLOG — Da el logaritmo de base 10 o logaritmo común.
Ej.: A = CLOG (B)

EXP — Da el valor de e (2.718 aproximadamente) elevado a la potencia especificada. En algunos casos, la precisión de EXP no abarca más que 6 dígitos significativos. Es el inverso de LOG.
Ej.: A = EXP (B)

INT — Da la integral mayor, menor o igual a un valor.
Ej.: A = INT (B)

LOG — Da el logaritmo natural de un valor. Es el inverso de EXP.

RND — Da un número aleatorio entre 0 y 1. Nunca da un 1. El valor no tiene importancia.
Ej.: A = RND (0) (A = un número igual a 0 o menor que 1)
A = RND (0)*8 (A = un número igual a o mayor que 0 e inferior a 8)

SGN — Da -1 si el valor es negativo, un cero si es 0, y un 1 si el valor es positivo.
Ej.: A = SGN (A)

SQR — Da la raíz cuadrada positiva de un valor positivo.
Ej.: A = SQR (B)

FUNCIONES TRIGONOMETRICAS

ATN — Da el arcotangente de un valor en radianes o grados.
Ej.: A = ATN (B)

SIN — Da el seno del valor.
Ej.: A = SIN (B)

COS — Da el coseno de un valor.
Ej.: A = COS (B)

DEG — Todas las funciones trigonométricas ulteriores serán expresadas en grados.
Ej.: DEG

RAD — Todas las funciones trigonométricas unteriores serán expresadas en radianes. El ordenador parte de la hipótesis de que se desea obtener radianes, salvo si especifica used DEG (grados).

TAN — Da el tangente de un valor.

FUNCIONES ESPECIALES

ADR — Da la dirección de memoria decimal del comienzo de una cadena.
Ej.: A = ADR (B$)
A = ADR ("THIS STRING") (esta cadena)

FRE — Da el número de bytes de RAM utilizador restantes. (0) es una variable "ficticia" exigida.
Ej.: A = FRE (0)
PRINT FRE (0) (lo que le permitirá saber cuánto le queda libre en la memoria).

PEEK — Da el número almacenado en un lugar específico de la memoria. La dirección debe consistir en un número comprendido entre 0 y 65535. El número dado estará comprendido entre 0 y 255.
Ej.: A = PEEK (B)

USR — Llama las subrutinas en lenguaje de máquina del BASIC. Por ejemplo: USR(ADDR, P1, P2, . . . PN), coloca los argumentos (de P1 a PN) en el lote de trabajo en orden inverso. Así pues, el último argumento "PN" se coloca en el lote antes del primer argumento "P1". Luego, el número de argumentos — representado por un solo byte — se agrega al lote. Si no se especifica ningún argumento en la función, se agrega cero al lote.
Entonces, se llama la subrutina en lenguaje de máquina a la dirección (ADDR). Si la rutina en lenguaje de máquina debe retornar un valor en BASIC, los bytes alto y bajo deben almacenarse en las direcciones de memoria $D4 y $D5 respectivamente. La rutina debe retirar la cuenta de argumentos y los argumentos en sí antes de volver al BASIC, de lo contrario el sistema entrará en crisis.
Ejemplo:
A = USR (B,C,D)
(Se llamará la rutina en B, y los parámetros en C y D serán transferidos a la subrutina a través del lote).

FUNCIONES DE CADENA

ASC — Da el número de código ATASCII para el primer carácter de una cadena.
Ej.: A = ASC ("A") (A será 65).
A = ASC (BS) (se pueden utilizar variables de cadenas).

CHR$ — Da el carácter representado por el número de código ATASCII especificado. Recíproco de ASC.
Ej.: AS = ASC$ (65) (A$ será "A").

LEN — Da un número que representa la longitud de una variable de cadena.
Ej.: A = LEN (AS)

STR$ — Da una cadena que representa un valor especifico (traduce un número en una cadena).
Ej.: A$ = STR$ (65) (A$ será igual a "65", que es una cadena).

VAL — Da una número que representa una cadena específica (traduce una cadena en un número).
Ej.: A = VAL ("100") (A será igual al número 100)

MANIPULACIÓN DE CADENAS

El ATARI BASIC no utiliza un formato en tabla de cadena para de cadenas. Un mando potente en medio de la cadena permite efectura la concatenación de cadenas y otros tipos de manipulación.
Ejemplos:
SUBSTRINGS (SUB CADENAS)
50 A$ = "DAVEMARKGRETCHEN"
60 B$ = A$ (9,16) (B$ es "GRETCHEN")

CONCANTENATION
(CONCATENACION)
50 A$ = "HI"
60 B$ = "FRED"
70 A$ (LEN (A$)+1) = B$ (A$ es "HI FRED")
SEARCHING A STRING (BUSQUEDA DE UNA CADENA)
50 FOR Z = 1 TO LEN (A$)
60 IF A$ (Z,Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z

FUNCIONES DEL MANDO DE JUEGO

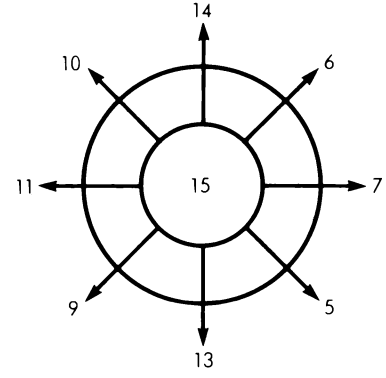
PADDLE — Da la posición de un mando específico. Los mandos están numerados de 0 a 3 desde delante hacia atrás. El número dado está entre 1 y 228, aumentando a medida que se gira el botón hacia la izquierda (en sentido contrario al de las manillas de un reloj).
Ej.: A = PADDLE (0)

PTRIG — Da un 0 si se pulsa un dispositivo de mando específico y un 1 en el caso contrario. Los mandos están numerados de l a 3, desde delante hacia atrás.
Ej.: A = PTRIG (0)

STICK — Da el estado de un mando de palanca específico. Los mandos de palanca están numerados 0 y 1, desde delante hacia atrás. Para detalles véase el diagrama.
Ej.: A = STICK (0)

STRIG — Da un 0 si se pulsa un disparador específico del mando de palanca, y un 1 en el caso contrario. Los mandos de palanca están numerados 0 y 1.
Ej.: A = STRIG (0).
Mando 1 = STICK (0)
Mando 2 = STICK (1)

El diagrama que sigue ilustra las cifras que son dadas cuando el mando de palanca es desplazado en una dirección cualquiera.



TECLAS PARA FUNCIONES ESPECIALES

ESC Hace que la tecla pulsada a continuación sea presentada en forma de carácter internacional o de carácter gráfico, y que sea ejecutada al tiempo en que aparece en la pantalla.

BREAK Interrumpe un programa BASIC.

RESET Detiene un programa que está siendo ejecutado, vuelve a poner la pantalla en modo gráfico 0, borra la pantalla y termina los archivos sin cerrarlos. No borra el programa.

SET-CLR-TAB Desplaza el cursor hasta la próxima posición de tabulación previamente determinada.

SHIFT-SET-CLR-TAB Elimina una tabulación.

CTRL SET-CLR-TAB Fija una tabulación.

CTRL 1 Conecta y desconecta el paso de la presentación en la pantalla.

CTRL 2 Hace funcionar la señal acústica.

CTRL 3 Indica el final de un archivo.

EDICION

SHIFT INSERT Inserción de una línea.

CTRL INSERT Inserción de un carácter.

DELETE BACK S Suprime el carácter que está a la izquierda del cursor y hace avanzar el cursor.

SHIFT DELETE BACK S Suprime una línea.

CTRL DELETE BACK S Suprime el carácter en que se halla el cursor y desplaza el resto de la línea para llenar el espacio vacío.

SHIFT CLEAR o CTRL CLEAR Borra la pantalla.

CRTL UP ARROW Desplaza el cursor hacia arriba.

CTRL DOWN ARROW Desplaza el cursor hacia abajo.

CTRL LEFT ARROW Desplaza el cursor hacia la izquierda.

CTRL RIGHT ARROW Desplaza el cursor hacia la derecha.

CODIGOS DE ERRORES

CÓDIGO DE ERROR	MENSAJE DE ERROR CORRESPONDIENTE
2	Memoria insuficiente
3	Error de valor
4	Demasiadas variables
5	Error en la longitud de cadena
6	Error ausencia de datos
7	Número superior a 32767
8	Error de instrucción de entrada
9	Error DIM de tabla o de cadena
10	Superación de pila de parámetros
11	Error de superación/ insuficiencia de coma flotante
12	Línea no hallada
13	Instrucción incompatible con FOR
14	Error de línea demasiado larga (superación de línea)
15	Línea GOSUB o FOR suprimida
16	Error de RETURN
17	Error de sintaxis
18	Cadena de caracteres inválida
19	El programa LOAD es demasiado largo.
20	El número de aparatos es mayor que 7.
21	Error de fichero LOAD.

NOTA: Los errores siguientes son errores de entrada/salida (INPUT/OUTPUT) que pueden presentarse cuando se utilizan unidades de discos, impresores o otros aparatos auxiliares. Junto con el material auxiliar se proporciona más información.

128	Abandono de BREAK
129	IOCB ya está abierto.
130	Aparato inexistente.
131	IOCB escritura (entrada) solamente.
132	Orden de manejo anulada.
133	Aparato o fichero cerrado.
134	Número Bad IOCB inválido.
135	Error IOCB de lectura sólo.
136	EOF-Final del Archivo.
137	Fichero truncado.
138	Interrupción del aparato.
139	Aparato NAK.
140	Serie bus.
141	Cursor fuera de alcance.
142	Superación trama de datos serie bus.
143	Error respecto a la suma de ensayo de la trama de datos, serie bus.
144	Error cometido por el aparato.
145	Error de elección de pantalla.
146	Función no realizada.
147	Memoria viva (RAM) insuficiente.
160	Error en el número de unidad del disco.
161	Exceso de ficheros OPEN (abiertos).
162	Disco completo.
163	Error de E/S, datos del sistema irrecuperables.
164	El número del fichero no corresponde.
165	Error del título del fichero.
166	Error respecto a la longitud de datos POINT.
167	Fichero cerrado.
168	Orden invalida del aparato.
169	Repertorio completo.
170	Fichero no hallado.
171	POINT invalidado.
172	Añadir archivo DOS I al archivo DOS II.
173	Sector erróneo en el formato.

NOTA: Le remotimos a la página 11 (ERROR CODES) de la sección en inglés para el mensaje de código de error tal como aparecere en la pantalla de su televisor.

PER PROGRAMMATORI ESPERTI

Imparare il BASIC è come imparare qualsiasi lingua — richiede un po' di tempo e un piccolo sforzo, ma se ne ricavano delle grosse soddisfazioni. Questo manuale fornisce informazioni relative all 'Atari BASIC — un linguaggio potente e di grande richiamo — per tutti quelli che hanno già una conoscenza del linguaggio di programmazione BASIC. Questo manuale è da utilizzare solo come riferimento. Non contiene esempi di programmazione globali nè informazioni didattiche per il principiante. Per ulteriori informazioni i programmatori, sia quelli esperti che i principianti, dovrebbero consultare le seguenti pubblicazioni: ATARI BASIC di Albrecht, Findel e Brown; ATARI BASIC REFERENCE MANUAL; e INSIDE ATARI BASIC di Bill Carris.

INDICE

PRECEDENZA DEL L'OPERATORE

Le operazioni contenute nei gruppi di parentesi più interni sono le prime a dover essere eseguite e vengono seguite da quelle del gruppo esterno. Quando un gruppo (di parentesi è contenuto in un altro gruppo) viene nominato "racchiuso". Le operazioni di uno stesso livello di raggruppamento sono eseguite nell' ordine seguente:

DALLA MASSIMA ALLA MINIMA PRECEDENZA

<, >, =, <=, >=, <>
Operatori relazionali usati nelle espressioni di stringa. Hanno la medesima precedenza e vengono eseguiti da sinistra a destra.

-
Negazione a complemento di area unitaria

Apposizione di esponenti.

***, /**
La moltiplicazione e la divisione hanno lo stesso livello di precedenza e vengono eseguite da sinistra a destra.

+, -
L'addizione e la sottrazione hanno lo stesso livello di precedenza e vengono eseguite da sinistra a destra.

<, >, =, <=, >=, <>
Le operazioni relazionali contenute in espressioni numeriche hanno lo stesso livello di precedenza da sinistra a destra.

NOT
Operatore a complemento di area unitaria

AND
AND logico

OR
OR logico

(Abbreviazioni accettate tra parentesi)
Gli Home Computer ATARI non fanno distinzione tra comandi e istruzioni. Le parole seguenti possono essere usate come istruzioni di programma oppure possono essere direttamente battute sulla tastiera seguita dal tasto RETURN. Queste parole non si possono usare come nomi di variabili.

CONTROLLO DEL SISTEMA

BYE (B.) — Esce dal modo BASIC e entra nel modo SELF TEST

DOS — Visualizza il menù DOS (da usare soltanto con unità a disco)

CSAVE (CS) — Salva il programma sulla cassetta

CLOAD — Carica il programma dalla cassetta

SAVE (S.) — Salva il programma BASIC su un 'unità di uscita
Es.: SAVE "D:MYFILE.BAS"

LOAD (LO.) — Carica un programma da un 'unità di entrata
Es.: LOAD "D: MYFILE.BAS"

LIST (L.) — Fa la lista di un programma sullo schermo o su un 'unità di uscita.
Es.: LIST (fa la lista dell 'intero programma)
List 10 (mostra la linea 10 sullo schermo)
LIST 10, 20 (fa la lista di tutto che si trova compresa tra le linee 10 e 20)
LIST "P:" (fa fare la lista alla stampante)
LIST "P:", 10, 20 (fa scrivere le linee 10, 20 sulla stampante)
LIST "D: MYFILE.LST" (fa fare la lista su un file su disco)
LIST "D: MYFILE.LST", 10, 20 (fa fare la lista delle linee 10, 20 su un file su disco)
LIST "C:" (fa fare la lista su cassetta)

ENTER (E.) — Carica il programma dal-l'unità di entrata.
Ex.: ENTER "C:"
ENTER "D: MYFILE.LST"
Le righe che hanno lo stesso numero in un programma che è già stato caricato verranno sovrascritte.

NEW — Cancella il programma dalla memoria.

RUN — Inizia l'esecuzione di un programma BASIC. Il programma può essere in memoria o può essere caricato da un disco o una cassetta. (Inizializza le variabili ponendole a zero e liberalizza il dimensionamento di matrici e stringhe).
Es.: RUN (esegue il programma in memoria)
RUN "D: MYFILE.BAS" (carica il programma dal disco e lo esegue)

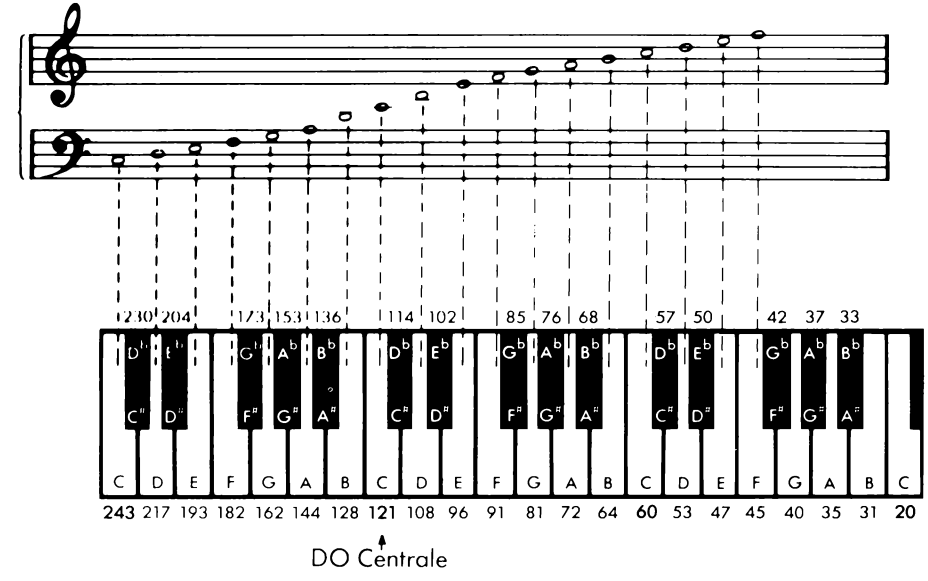
CONT — Continua l'esecuzione del programma dopo che il tasto di interruzione è stato premuto o dopo che il programma ha eseguito un 'istruzione di STOP o END. Se sulla stessa linea ci sono istruzioni addizionali di programma, queste non vengono eseguite. Il programma prosegue l'esecuzione con la linea numerata successiva.

ISTRUZIONI DEL SUONO

SOUND (SO.) — Predispone uno dei quattro canali alla produzione del suono attraverso l'audio TV. Il suono continua fino a che un 'altra istruzione SOUND non indirizzi lo stesso canale o non venga eseguita un 'istruzione di END, RUN o NEW. I canali possono essere programmati indipendentemente e avere funzionamento simultaneo. Questa instruzione deve essere seguita da quattro valori (numeri, variabili o espressioni).

Es.: SOUND A, B, C, D dove:
A = numero del canale (0-3)
B = periodicità (0-225). Più grande è il valore, più bassa è la frequenza. (Frequenza = 31960/PERIOD + 1) Vedi figura per equivalenti musicali.
C = Distorsione (0-14, solo # pari) Dieci e 14 sono toni "puri". Altri numeri risultano in suoni diversi.
D = Volume (0-15). Più il valore è grande, più il suono è forte. Lo zero corrisponde all'assenza di suono. Se il volume totale per tutte e 4 le voci è superiore a 32, l'altoparlante può emettere un segnale di avvertimento (cicalino).

POSIZIONE DELLA TASTIERA RISPETTO ALLA SCALA MUSICALE



CONTROLLO DEL PROGRAMMA

GOTO (G.) — L'esecuzione del programma continua al numero di linea specificato
Es.: GOTO 30 (esegue il programma a partire dalla linea 30)
Es.: GOTO A + 10 (è accettato ma complica il debugging)

ON. . .GOTO — L'esecuzione del programma continua al numero di linea indicato da un 'espressione
Es.: 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.) — L'esecuzione del programma continua al numero di linea indicato. L'istruzione RETURN fa ritornare il controllo all'instruzione che segue la GOSUB
Es.: GOSUB 30 (eseguire il sottoprogramma alla riga 30)
GOSUB A+ 10 (è permesso ma può complicare il debugging)

ON. . .GOSUB — L'esecuzione del programma continua al numero di linea indicato da un' espressione. RETURN fa ritornare il controllo all' istruzione che segue la ON. . .GOSUB
Es.: 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.)
Attenzione: se l'espressione valuta un numero come più piccolo di 1 o più grande del numero di numeri di riga, i risultati sono imprevedibili.

RETURN (RET.) — Termina un sottoprogramma e fa ritornare il controllo all'istruzione immediatamente seguente l'ultima istruzione GOSUB eseguita.
Es.: RETURN.

FOR (F.) — Questa istruzione fissa i valori iniziale e finale di un indice variabile e il valore incrementale da sommare all' indice ogni volta che viene eseguito il ciclo FOR. . .NEXT. L'incremento è 1 a meno che un valore diverso sia stato specificato con istruzione STEP. Un 'istruzione NEXT fa ripetere le istruzioni tra FOR e NEXT.
Es.: FOR A = 1 TO 10 (il valore iniziale di A è 1, l'incremento è 1. Il ciclo si chiude quando A raggiunge il valore 10)
FOR A = 10 TO 1 STEP-2 (A meno che STEP sia diverso da 1, cioè - 2)
FOR A = B/T TO B*T STEP X (sono validi anche i valori calcolati)

NEXT (N.) — Termina un ciclo FOR . . . NEXT. Controlla che il valore dell' indice non abbia superato il valore finale, incrementa il valore dell' indice del valore di STEP e continua l'esecuzione dell' istruzione che segue la FOR. Quando l'indice supera il valore finale, continua alla istruzione che segue la NEXT.
ES.: NEXT A (A è l'indice variabile)

POP — Annulla i dati di ritorno relativi all 'ultima istruzione FOR o GOSUB eseguita. E' utile per uscire velocemente da un ciclo FOR-NEXT, oppure per lasciare un sottoprogramma senza passare per l'istruzione RETURN.
ES.: POP: GOTO 10

IF. . .THEN — L'istruzione specificata dopo il termine THEN viene eseguita se la condizione tra IF e THEN è vera. In caso contrario il programma continua alla linea successiva.
Es.: IF A ≠ C THEN GOTO 300
IF A = B THEN PRINT
"A = B": PRINT
"Che le sembra"
LET A = 5: GOTO 20
IF A THEN PRINT "A è diverso da zero"
(un espressione diversa da zero è vera)

TRAP (T.) — In caso di errore TRAP fa continuare l'esecuzione alla linea specificata. Questa linea rimane valida finchè non si verifica un errore di esecuzione oppure fino alla esecuzione della prossima istruzione TRAP. PEEK (195) restituisce il numero dell 'errore. PEEK (187.256 + PEEK (186) restituisce il numero della linea.
Es.: TRAP 30 (in caso di errore continuare alla linea 30)
TRAP 40000 (i numeri di linea maggiori di 32767 annullano l'effetto di TRAP)

STOP — Ferma il programma. Stampa il numero della linea. Non chiude i file nè spegne il suono. Si può ricominciare un programma con CONT.
Es.: STOP

END — Ferma il programma, chiude tutti i file aperti, spegne il suono. Si può ricominciare il programma con CONT.

ENTRATA E USCITA

NOMI DELLE UNITÀ DI I/O

Ogni unità della famiglia dei prodotti ATARI ha un nome esclusivo. Le unità a disco e il modulo di interfaccia ATARI 850™ (RS232) hanno bisogno del numero dell'unita (1-4). Le unità a disco hanno bisogno anche del nome dell 'unità. I nomi delle unità devono essere compresi o contenuti in variabili di stringa. Ecco alcuni esempi:

K: Tastiera. Solo entrata
P: Stampante. Solo uscita
C: Cassetta. entrata e uscita
S: Schermo (TV). Solo uscita
E: Editore schermo (tastiera abbinata a schermo). Entrata e uscita
R: RS232 (modulo interfaccia ATARI 850)

D:FILENAME.EXT	Eseguita il File: "FILENAME.EXT sul disk drive #1"
D2:FILENAME.EXT	Stesso file sull 'unita #2

ISTRUZIONI DI I/O

I nomi di file su disco incominciano con una lettera e possono avere fino a 8 caratteri. Il nome di un file può eventualmente essere prolungato con un 'estensione (un punto seguito da l a 3 caratteri qualsiasi, numerici o alfanumerici). Si può usare 3 lettere o numeri qualunque che si vuole. Alcune estensioni utili sono:

.BAS = Conserva i programmi BASIC (come SAVE)
.LST = Elenca i programmi BASIC (come LIST)
.DAT = file di dati
.OBJ = file del linguaggio macchina (oggetto)
.TXT = file del testo

OPEN (O.) — Prepara un' unità per l'entrata o l'uscita. I numeri IOCB sono 1-7. Usa i seguenti codici:

TIPO	CODICE	OPERAZIONE
Entrata	4	Solo lettura
Uscita	8	Solo scrittura
Aggiornamento	12	Lettura e scrittura
Aggiunta	9	Accoda a partire dalla fine del file
Indice	6	Operazioni sull'indice dell'unità a disco

Es.: OPEN # 1,4,0,"K": (apre la tastiera per entrata su IOCB#1.)
OPEN #2,8,0, "P" (apre la stampante per uscita du IOCB #2)
Open #1, 12, 0, "D: MYFILE.DAT" (apre il file su disco "MYFILE.DAT" per aggiornamento su IOCB #1)
OPEN #1,6,0,"D:*.*" (apre 1' unità a disco 1 per operazioni sull 'indice su IOCB #1.)

CLOSE (CL:) — Chiude l'unità dopo l'operazione di entrata o uscita e libera lo IOCB. Si può eseguire quando nessun'unità è stata aperta. Gli IOCB sono 1-7 come in OPEN.

Es.: CLOSE #1 (chiude il file aperto su IOCB #1 e libera l'IOCB)

INPUT (I.) — Legge una linea di caratteri dall 'unità periferica. La linea deve essere terminata con un carattere RETURN.

Es.: INPUT A (legge un numero e lo mette in A)
INPUT A, B, C (legge 3 numeri, separati da virgole, e li mette in A, B & C)
INPUT A$ (legge una linea di caratteri e li mette in A$, A$ non conterrà il carattere RETURN)
INPUT #1, A$, B (legge una linea dal l'unità aperta su IOCB #1 e la mette in A$ & B)

PRINT (PR.) oppure **(?)** — Invia i dati allo schermo o ad un' altra unità periferica

Es.: PRINT (invia una linea bianca)
PRINT "Il numero è", A (stampa testo e numero sullo schermo)
PRINT "Il numero è", A (il numero è, A) (La virgola provoca la stampa di A in una colonna separata. POKE 201 con la larghezza di colonna desiderata.)
PRINT A$; (il punto e virgola impedisce che il carattere RETURN venga trasmesso alla fine della linea)
PRINT 1; A$ (invia A$ all 'unità aperta su IOCB #1)

LPRINT (LP.) — Stampa i dati sulla stampante. Non è necessaria nessuna operazione di apertura o chiusura. Il punto e virgola alla fine della riga non impedisce che venga inviato il carattere RETURN.

Es.: LPRINT (invia una linea bianca alla stampante)
LPRINT A$ (stampa A$)
LPRINT A$,B (A$ e B saranno sulla stessa riga)
LPRINT A$,B (La virgola provoca la stampa di B in una colonna separata. POKE 201 con la larghezza di colonna desiderata.)

GET — Legge un singolo byte dall 'unità periferica specificata e lo mette nella variabile specificata)

Es.: GET #1,A (Legge un byte dall 'unità aperta su IOCB #1 e lo mette in A)

PUT — Invia un singolo byte della variabile all 'unità periferica specificata

Es.: PUT # 1,A (invia il byte in A all'unità aperta sull 'IOCB #1)

NOTE (NO.) — Usata con l'unità a disco per determinare la posizione del prossimo byte da leggere o scrivere

Es.: NOTE #1, SEC, BYTE (invia il numero del settore in SEC e il numero del byte in BYTE. Si referisce al file aperto su IOCB #1)

POINT (P.) — Usato per dire a DOS a quale punto leggere o a quale punto scrivere il prossimo byte

Es.: POINT #1, SEC, BYTE (indirezza DOS al numero di settore SEC e al numero di byte BYTE)

STATUS (ST.) — Legge lo stato del l'unità periferica specificata. Il codice di stato restituito si può trovare nella lista dei messaggi di errore.

Es.: STATUS #1, A (mette in A lo stato dell 'unità aperta su IOCB #1)

ISTRUZIONI DI CALCOLO

LET — Assegna valori a variabili numeriche o a variabili di stringa
Es.: LET A = B (valori di B assegnato ad A)
LET A$ = "HELLO" (salve)
A = B:A$ = "HELLO" (LET si può omettere)

POKE — Mette un numero compreso tra 0 e 255 in una posizione specificata di memoria compresa tra 0 e 65535, I valori decimali vengono arrotondati. PEEK preleva dalla memoria.
Es.: POKE 82,0 (mette 0 nella posizione di memoria 82) A = PEEK (82) (preleva il contenuto della posizione di memoria 82 e lo mette in A)

DIM — Riserva spazio in memoria per stringhe e tabelle di numeri. Ogni spazio riservato per una stringa richiede un byte; ogni elemento in una tabella numerica richiede sei byte.
Es.: DIM A$ (10) (una variabile di stringa)
DIM B (10) (una tabella numerica; B contiene gli elementi 0-10)
DIM B (10,10) (tabella numerica a due dimensioni)
DIM A$ (10), B (10) (si usi la virgola come separatore)

COM — Uguale a DIM

CLR — Cancella le dimensioni di qualsiasi array, e cancella stringhe e riduce le variabili numeriche a zero.
Es.: CLR

DATA (D.) — Crea una lista di numeri e/o lettere da usare con l'istruzione READ qui di seguito
Es.: DATA 1,2,3,4,A,B,C,D,(lista d'informazioni da leggere)

READ — Legge l'elemento successivo di un 'istruzione DATA e lo assegna a una variabile. Quando la lista di un 'istruzione DATA è stata esaurita, READ riceve i dati dall 'istruzione DATA seguente nel programma
Es.: READ A (A sarà il numero seguente nella lista dell 'istruzione DATA)
READ A$ (vale anche per stringhe)
READ A, A$, B, B$ (separare gli elementi di una lista con virgole)

RESTORE (RES.) — Rinvia la READ a un'istruzione DATA
Es.: RESTORE (Il prossimo campo di dati sarà il primo campo della prima istruzione DATA)
RESTORE 10 (il prossimo campo di dati sarà il primo elemento che si trova sulla lista dell' istruzione DATA alla linea 10)

REM (R.) o ([SPAZIO].) — Permette di inserire osservazioni.
BASIC ignora tutto da REM alla fine della linea.
Es.: REM (Questa è un 'istruzione di commento)

COMPOSIZIONI GRAFICHE

GRAPHICS (GR.) — Sceglie il modo 'espressione graficà. Il modo + 16 sceglie lo schermo pieno (niente finestra per il testo). La scelta del modo + 32 non azzera lo schermo. Modi 0-15 disponibile.
Es.: GRAPHICS 8 (modo grafico 8 con finestra)
GRAPHICS 8 + 16 (modo 8, schermo pieno)
GRAPHICS 8 + 32 (non azzera lo schermo)
GRAPHICS 8 + 16 + 32 (entrambe le possibilità combinate)

SETCOLOR (SE.) — Determina la sfumatura e la luminosità di un registro di colore prescelto. Il numero del registro non è lo stesso che per il comando COLOR.
Es.: SETCOLOR 1,2,4 (posiziona il registro 1 su sfumatura 2 e luminosità 4)
Registri 0-4
Sfumature 0-15
Luminosità 0-14, solo numeri pari (eccetto i modi GTIA, che fanno uso di numeri pari e dispari fino a 15)

COLOR (C.) — Nei modi operativi 3-11 (mappa) sceglie un registro di colore da usare per PLOT. Il registro qui non è lo stesso che quello usato da SETCOLOR.
Es.: COLOR 2 (scelglie il registro di colore 2 nei modi 0-2 sceglie il carattere ASCII il cui valore per PLOT è 2)

TABELLA DEI MODI E DEI FORMATI DELLOO SCHERMO

FORMATO DELLO SCHERMO

Modo Grafico	Tipo	Colonne	Righe- (schermo diviso)	Righe- (schermo pieno)	Numero colori	Memoria RAM richiesta (Bytes) divisa	piena
0	TEXT	40	—	24	1-1/2	—	992
1	TEXT	20	20	24	5	674	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

PLOT (PL.) — Mette un singolo punto o carattere sullo schermo in un punto specificato.
Es.: PLOT X, Y (coordinate X e Y devono avere valor: positivi.)

POSITION (POS.) — Sceglie una posizione sullo schermo senza tuttavia tracciare alcunchè. Utile per posizionare il testo col comando PRINT.
Es.: POSITION X, Y

LOCATE (LOC.) — Localizza i dati che si trovano in un punto specificato dello schermo. Nei modi 0-2 legge i caratteri, nei modi 3-11 legge i numeri dei colori.
Es.: LOCATE X,Y,D (si posiziona a X, Y, e mette i dati in D)

DRAWTO (DR.) — Traccia una linea tra l'ultima posizione del cursore e le coordinate specificate X e Y. Il cursore assume la posizione delle nuove coordinate.
Es.: DRAWTO X, Y (traccia una linea dalla posizione corrente del cursore al punto x, y)

SCHEMA DEI COLORI ADOTTATI PER DIFETTO

Setcolor (Registro Colore)	Colore Adottato per difetto	Lumi nosità	Colore effettivo
0	2	8	arancio
1	12	10	verde
2	9	4	blu scuro
3	4	6	rosa o rosso
4	0	0	nero

Il valore per difetto viene adottato se non viene usata l'istruzione SETCOLOR

Osservazione: I colori possono variare a seconda del tipo, condizioni e regolazione del televisore

I NUMERI DI COLORI E SFUMATURE ATARI (COMANDO SETCOLOR)

Colori	Numeri di colore Setcolor (aexp2)
GRIGIO	0
ARANCIO CHIARO (ORO)	1
ARANCIO	2
ROSSO ARANCIO	3
ROSA	4
PORPORA	5
PORPORA-BLU	6
BLU	7
BLU	8
AZZURRO	9
TURCHESE	10
VERDE-BLU	11
VERDE	12
GIALLO-VERDE	13
ARANCIO-VERDE	14
ARANCIO CHIARO	15

Note: Colors vary with type and adjustment of TV or monitor used.

FUNZIONI

Una funzione prende uno o più valori e restituisce un altro valore. I valori possono essere stringhe o numeri. Gli esempi dimostrano A (variabile numerica) o A$ (variabile di stringa) si possono considerare come uguale ai funzioni, ma una funzione si può usare quasi ovunque dove si userebbe un valore, perfino in un'altra funzione.

FUNZIONI ARITMETICHE

ABS — Ritorna il valore assoluto (senza segno) di un numero
Es.: A = ABS (B)

CLOG — Ritorna il logaritmo in base 10
Es.: A = CLOG (B)

EXP — Ritorna il valore di "e" (circa 2,718) elevato alla potenza specificata. In alcuni casi EXP da solo 6 cifre precise. Questo è l'opposto di LOG.
Es.: A = EXP (B)

INT — Ritorna il più grande numero intero minore o uguale a un dato valore.
Es.: A = INT (B)

LOG — Ritorna il logaritmo naturale di un numero. Questo è l'opposto di EXP.

RND — Ritorna un numero a caso compreso tra 0 e 1. Non ritorna mai il numero 1. Il valore dell 'argomento della funzione non influisce sul risultato.
Es.: A = RND (0) (A = numero uguale o maggiore di 0 e minore di 1)
A = RND (0) * 8 (A = numero uguale o maggiore di 0 e inferiore a 8)

SGN — Ritorna -1 se il valore è negativo, zero se il valore è nullo e I se il valore è positivo.
Es.: A = SGN (A)

SQR — Ritorna la radice quadrata positiva di un numero positivo.
Es.: A = SQR (B)

FUNZIONI TRIGONOMETRICHE

ATN — Ritorna l'arco tangente di un valore in radianti o gradi
Es.: A = ATN (B)

COS — Ritorna il coseno di un numero
Es.: A = COS (B)

SIN — Ritorna il seno di un valore
Es.: A = SIN (B)

DEG — Tutte le funzioni trigonometriche seguenti saranno in gradi
Es.: DEG

RAD — Tutte le funzioni trigonometriche seguenti saranno in radianti. Il computer suppone che vogliate il valore esperesso in radianti a meno che non precisiate DEG (gradi)

TAN — Da il tangente di un valore.

FUNZIONI SPECIALI

ADR — Ritorna l'indirizzo decimale di memoria del primo carattere di una stringa
Es.: A = ADR (B$)
A = ADR ("THIS STRING") (questa stringa)

FRE — Ritorna il numero di byte della memoria RAM che ancora rimangono all'utente. (0) è una variabile fittizia che deve essere specificata.
Es.: A = FRE (0)
PRINT Fre (o) (Per sapere quanto memoria rimane.)

PEEK — Ritorna il numero memorizzato a un dato indirizzo della memoria. L'indirizzo deve essere un numero compreso tra 0 e 65535. Il valore del numero ritornato è compreso tra 0 e 255.
Es.: A = PEEK (B)

USR — Richiama delle subroutines in linguaggio macchina dal BASIC. Per esempio: USR(ADDR, P1, P2,...,PN), spinge gli argomenti (da P1 a PN) sulla coda di lavoro in ordine inverso. Quindi L'ultimo argomento "PN" viene spinto sulla coda prima del primo argomento "P1"
Il numero di argomenti—rappresentato come byte singolo—viene poi spinto sulla coda. Se nessun argomento è specificato nella funzione, zero viene spinto sulla coda.
Viene poi richiamata la subroutine in linguaggio macchina all'indirizzo (ADDR). Se il programma in linguaggio macchina deve rinviare un valore in BASIC, i bytes alto e basso devono essere memorizzati rispettivamente negli indirizzi di memoria $D4 e $D5.
Il programma deve rimuovere il conteggio degli argomenti e gli argomenti stessi prima di ritornare al BASIC, altrimenti il sistema entrerà in crisi.
Esempio: A=USR(B,C,D)
(Verrà richiamato il programma a B, ed i parametri in C e D verranno passati alla subroutine tramite la coda).

FUNZIONI DI STRINGA

ASC — Ritorna il codice ATASCII corrispondente al primo carattere di una stringa
Es.: A = ASC ("A") (A sarà 65)
A = ASC (B$) (si possono usare variabili di stringa)

CHR$ — Ritorna il carattere corrispondente al numero di codice ATASC II specificato. E' l'inverso di ASC.
Es.: A$ = A$C (65) (A$ sarà "A")

LEN — Ritorna un numero che rappresenta la lunghezza di una variabile di stringa
Es.: A = LEN (A$)

STR$ — Ritorna una stringa che corrisponde a un valore specificato (traduce un numero in una stringa)
Es.: A$ = STR$ (65) (A$ sarà pari a "65" che è una stringa)

VAL — Ritorna un numero corrispondente a una stringa (traduce una stringa in un numero)
Es.: A = VAL ("100") (A sarà uguale al numero 100)

MANIPOLAZIONE DI STRINGHE

L'ATARI BASIC non si serve di formati per la manipolazione di stringhe. Un potente comando su sottostringhe agevola le operazione di concatenazione, ricerca e altre.
Esempi:
SOTTOSTRINGHE
50 A$ = "DAVEMARKGRETCHEN"
60 B$ = A$(9,16)(B$ is "GRETCHEN")

CONCATENAZIONE
50 A$ = "HI"
60 B$ = "FRED"
70 A$(LEN(A$)+1) + B$(A$ is "HI FRED")

RICERCA DI UNA STRINGA
50 FOR Z = 1 TO LEN(A$)
60 IF A$(Z, Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z

FUNZIONI DI CONTROLLO DEI GIOCHI

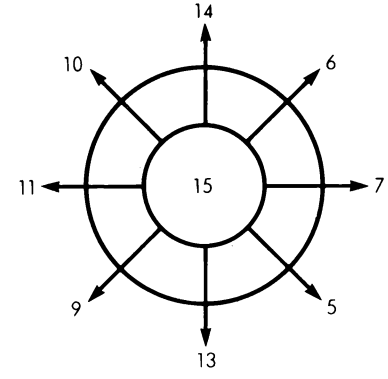
PADDLE — Ritorna la posizione della manopola di controllo delle racchette. Le racchette sono numerate da 0 a 3 dal davanti verso il retro. Il numero ritornato è compreso tra 1 e 228, e aumenta quando la manopola viene vuotata verso sinistra (in senso antiorario)
Es.: A = PADDLE (0)

PTRIG — Ritorna 0 se viene premuto un comando specifico di racchetta e l se non viene premuto. Le racchette sono numerate da 0 a 3 dal davanti verso il retro.
Es.: A = PTRIG (0)

STICK — Ritorna lo stato di un comando a cloche. Le cloche sono numerate 0-1 dal davanti verso il retro. Vedere il diagramma per i dettagli
Es.: A = STICK (0)

STRIG — Ritorna uno 0 se viene premuto un pulsante specifico di cloche e un l se non viene premuto. Le cloche sono numerate 0 e 1
Es.: A = STRIG (0)
Comando 1 = STICK (0)
Comando 2 = STICK (1)

Il seguente diagramma mostra i numeri di ritorno quando il comando a cloche viene spostato nelle varie direzioni.



TASTI DI FUNZIONI SPECIALI

ESC Un tasto premuto subito dopo l'ESC viene visualizzato come carattere internazionale o carattere grafico ed eseguito quando viene inviato sullo schermo

BREAK Ferma l'esecuzione di un programma BASIC

RESET Ferma un programma in esecuzione, rimette lo schermo nel modo grafico 0, azzera lo schermo e termina i vostri file senza chiuderli. Non cancella il programma.

SET-CLR-TAB Muove il cursore al successivo arresto del tabulatore

SHIFT SET-CLR-TAB Annulla una posizione del tabulatore

CTRL 1 Ferma e inizia lo scorrimento dello schermo

CTRL 2 Fa suonare il cicalino

CTRL 3 Indica la fine di un file

DIRIGENDO

SHIFT INSERT Inserisce una linea

CTRL INSERT Inserisce un carattere

DELETE BACK S Cancella il carattere a sinistra del cursore e sposta il cursore

SHIFT DELETE BACK S Cancella una linea

CTRL DELETE BACK S Cancella il carattere che si trova in corrispondenza del cursore e sposta il resto della linea per riempire la posizione vuota

SHIFT CLEAR or **CTRL CLEAR** Azzera lo schermo

CTRL UP ARROW Muove il cursore in su

CTRL DOWN ARROW Muove il cursore in giù

CTRL LEFT ARROW Muove il cursore a sinistra

CTRL RIGHT ARROW Muove il cursore a destra

(**HELP**) Inizia l'auto-test. Può essere programmato per fornire suggerimenti

(**CAPS**) Fa passare da maiuscole a minuscole e viceversa

CODICI DI ERRORE

CODICE DI ERRORE	MESSAGIO DI ERRORE CORRISPONDENTE
2	Memoria insufficiente
3	Errore di valore
4	Troppe variabili
5	Errore di lunghezza di stringa
6	Dati insufficienti
7	Numero superiore a 32767
8	Errore d'istruzione di entrata
9	Errore DIM di tabella o di stringa
10	Supero di capacità di argomenti impilati
11	Supero di capacità per valori troppo grandi o troppo piccoli in virgola mobile
12	Linea inesistente
13	Manca l'istruzione corrispondente FOR
14	Linea troppo lunga
15	Cancellata linea GOSUB o FOR
16	Errore di RETURN
17	Errore di sintassi
18	Carattere di stringa non valido

Osservazione: Seguono gli errori di entrata/uscita che si verificano durante l'uso di unità a disco, stampanti o altre unità periferiche. Informazioni ulteriori sono fornite col hardware ausiliario

19	Programma LOAD troppo lungo
20	Numero di unità maggiore di 7
21	Errore sul file LOAD
128	Interruzione forzata di BREAK
129	IOCB già aperto
130	Unità inesistente
131	IOCB valido per sola scrittura
132	Comando non valido per l'attuatore
133	Unità o file non aperto
134	Numero di IOCB incorretto
135	IOCB valido per sola scrittura
136	EOF
137	Registrazione troncata
138	Errore di disincronizzazione di unità periferica
139	NAK (Unità periferica non identificata)
140	Bus seriale
141	Cursore fuori intervallo
142	Distruzione di un blocco dati nel bus seriale a causa di diversità nella velocità di trasferimento
143	Errore di checksum di un blocco dati nel bus seriale
144	Errore di unità periferica
145	Errore di selezione di schermo
146	Funzione inesistente
147	RAM insufficiente
160	Numero di unità periferica incorretto
161	Troppi file aperti
162	Disco pieno
163	Errore di I/O irrecuperabile
164	Numero del file non corrisponde
165	Nome di file incorretto
166	Errore di lunghezza in un'istruzione POINT
167	Accesso al file bloccato
168	Comando non valido per l'unità periferica
169	Indice completo
170	File inesistente o non identificato
171	POINT non valido
172	Accodamento del file DOS I al file DOS II
173	Identificato settore disco difettoso nella fase di formattazione

Nota: Vedi pagina 11 (ERROR CODES) nella sezione inglese per la lista dei codici di errore come appariranno su vostro schermo.

POUR PROGRAMMEURS EXPERIMENTES

Apprendre le BASIC, c'est un peu comme apprendre une langue étrangère: il faut y consacrer un peu de temps et quelques efforts, mais la récompense en vaut la peine. Ce manuel vous donnera un certain nombre de renseignements sur le BASIC ATARI — un langage simple et puissant —destiné à ceux qui désirent programmer en BASIC.

Ce manuel est un guide de référence. Il ne présente pas une série complète d'exemples de programmation et ne constitue pas un manuel d'instruction complet à l'usage du débutant. Nous conseillons aux lecteurs non familiarisés avec le langage BASIC de se procurer un manuel destiné aux débutants; il en existe beaucoup d'excellents sur le marché.

INDEX

ORDRE DE PRIORITE DES OPERATEURS

Les opérations qui sont insérées dans les parenthèses les plus à l'intérieur doivent être exécutées en premier. Lorsqu' un jeu de parenthèses est compris à l'intérieur d'un autre jeu de parenthèses, on parle de parenthèses "imbriquées". Les opérations qui sont au même niveau s'exécutent dans l'ordre suivant:

DE LA PRIORITE LA PLUS ELEVEE A LA PRIORITE LA PLUS BASSE

<, >, =, <=, > =, < >
Opérateurs relationnels utilisés dans les expressions chaîne de caractères. Ils ont le même ordre de priorité et sont exécutés de gauche à droite.

-
Signe moins unaire, z. B.: x=-1

Elévation à une puissance

***, /**
Multiplication et division ont le même ordre de priorité et s'exécutent de gauche à droite.

+, -
Addition et soustraction ont le même ordre de priorité et s'exécutent de gauche à droite.

<, >, =, <=, > =, < >
Opérateurs relationnels utilisés dans les expressions numériques

NOT
NON logique

AND
AND (ET) logique

OR
OR (OU) logique

(Abréviations permises entre parenthèses)
Les mots qui suivent ne peuvent pas être utilisés comme noms de variables. La majorité de ces instructions peut être tapée directement au clavier ou être utilisée à l'intérieur d'un programme.

COMMANDE DU SYSTEME

BYE (B.) — Sort du mode BASIC pour passer en mode SELF TEST.

DOS (DO.) — Affiche le menu DOS (à utiliser avec une unité de disquette seulement)

CSAVE (CS.) — Sauvegarde le programme sur cassette.

CLOAD (CLOA.) — Charge le programme dans la mémoire de l'ordinateur à partir d'une cassette.

SAVE (S.) — Sauvegarde un programme en BASIC, sur une disquette ou tout autre périphérique.
Ex.: SAVE "D:FICHIER.BAS"

LOAD (LO.) — Charge un programme dans la mémoire de l'ordinateur à partir d'un périphérique d'entrée.
Ex.: LOAD "D:FICHIER.BAS"

LIST (L.) — Liste un programme à l'écran ou sur un périphérique.
Ex.: LIST (liste la totalité du programme).
LIST 10 (liste la ligne 10 à l'écran)
LIST 10,20 (liste tout ce qui se trouve de la ligne 10 à la ligne 20 incluse)
LIST "P:" (liste à l'imprimante)
LIST "P:", 10, 20 (les lignes 10 à 20 sont listées sur l'imprimante)
LIST "D FICHIER" (liste à un fichier sur disquette)
LIST "D: FICHIER.LST" 10,20 (liste les lignes 10 à 20 dans un fichier sur disquette).
LIST "C:" (liste sur cassette).

ENTER (E.) — Entre le programme dans l'ordinateur.
Ex.: ENTER "C:"
ENTER "D: FICHIER.LST"
Note: Les lignes de même numéro d'un programme déjà chargé seront remplacées.

NEW — Efface tout programme se trouvant dans la mémoire de l'ordinateur.

RUN (RU.) — Lance l'exécution d'un programme en BASIC. Le programme peut se trouver en mémoire ou bien être chargé à partir d'une disquette ou d'une cassette. Initialise les variables en les remettant à zéro et supprime les tableaux et les chaînes de caractères.
Ex.: RUN (exécute le programme en mémoire).
RUN "D: FICHIER.BAS" (CHARGE le programme à partir de la disquette et l'exécute).

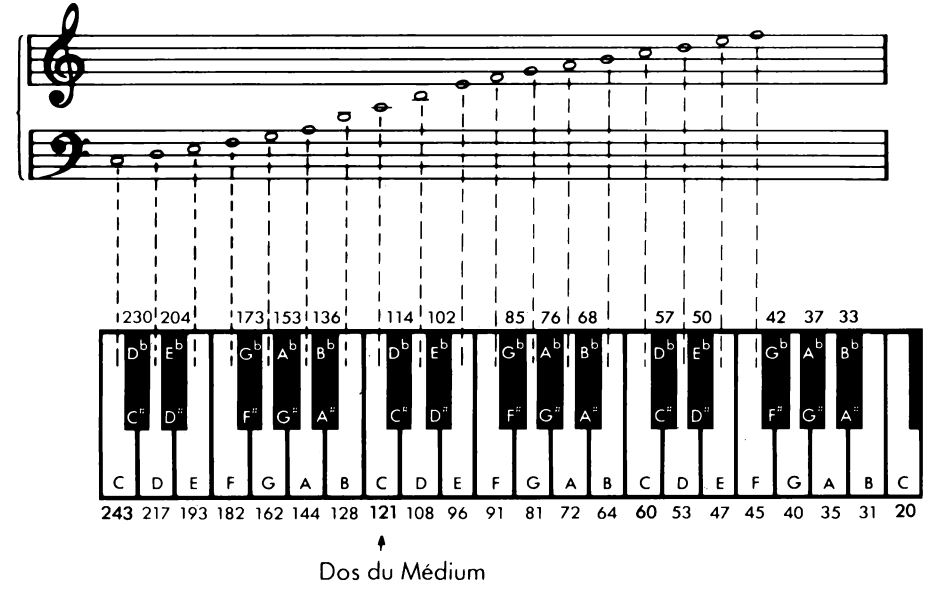
CONT — Reprend l'exécution du programme après que la touche BREAK ait été pressée ou que le programme ait exécuté une instruction STOP ou END. S'il existe d'autres instructions de programme sur la même ligne, celles-ci ne sont pas exécutées. L'exécution du programme reprend à la ligne numérotée suivante.

INSTRUCTIONS SONORES

SOUND (SO.) — Provoque l'émission d'un son. L'ordinateur ATARI peut jouer jusqu'à quatre sons simultanément. On parle alors de voix. Cela correspond à l'intérieur de l'ordinateur à quatre synthétiseurs indépendants. Le son dure jusqu'à ce qu'une autre instruction SOUND soit adressée à la même voix, ou qu'une instruction END, RUN ou NEW soit exécutée. Les voix sont programmées indépendamment et peuvent toutes fonctionner simultanément. Cette instruction doit être suivie par quatre valeurs (nombres, variables ou expressions).

Ex.: SOUND A, B, C, D où:
A = Numéro de canal (0 à 3)
B = Période (0 à 255). Plus la période est grande, plus la fréquence est basse. Fréquence = 31960/PERIODE + 1). Voir le tableau pour les équivalences musicales.
C = Distorsion (0 à 14, valeurs paires seulement). 10 et 14 sont des tons "purs". Les autres nombres donnent d'autres sonorités.
D = Volume (0 à 15). Plus le nombre est grand, plus le volume sonore est fort. Le chiffre 0 ne produit aucun son. Il se peut que le son soit très fortement distordu si le volume total des quatre voix dépasse 32.

CORRESPONDANCE ENTRE LE CLAVIER DU PIAN ET LA GAMME MUSICALE



COMMANDES DES PROGRAMMES

GOTO (G.) — L'exécution du programme se fait à partir du numéro de ligne qui a été spécifié.
Ex.: GOTO 30 (exécute le programme à partir de la ligne 30).
GOTO A + 10 (autorisé, mais peut être difficile à déterminer).

ON. . .GOTO — L'exécution du programme passe à l'un des numéros de ligne spécifié après le mot GOTO. Le choix s'effectue selon la valeur de l'expression suivant ON.
Ex.: 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.) — L'exécution du programme continue à la ligne spécifiée. Une instruction RETURN fait revenir le programme à l'instruction qui suit ON . . . GOSUB.
Ex.: GOSUB 30 (executer sous-programme à la ligna 30)
GOSUB A + 10 (légal, mais peut s'avérer tres difficile à mettre au point).

ON . . . GOSUB — L'exécution du programme continue au numéro de ligne indiqué par une expression. Une pression de RETURN fera revenir la commande à l'instruction qui suit le ON . . . GOSUB
Ex.: ON A + 1 GOSUB 10, 300, 50
(IF A + 1 = 1 THEN GOSUB 10;
IF A + 1 = 2, THEN GOSUB 300;
I F A + 1 = 3, THEN GOSUB 50).
Attention: si l'expression évalue un nombre inférieur à 1 ou supérieur au numéro de ligne le plus élevé, les résultats sont imprévisibles.

RETURN (RET.) — Termine un sous-programme et ramène le programme à l'instruction qui suit immédiatement la dernière instruction GOSUB exécutée.
Ex.: RETURN

FOR (F.) — Cette instruction fixe les valeurs de départ et de fin d'une variable d'index, ainsi que la valeur qui doit lui être ajoutée chaque fois qu'une boucle FOR. . .NEXT s'exécute. La valeur ajoutée est 1 à moins qu'il n'en soit spécifié autrement par une instruction STEP. Une instruction NEXT fera se rèpèter l'instruction entre FOR et NEXT.
Ex.: FOR A = 1 TO 10 (A commencera à 1 et augmentera chaque fois de 1 pour s'arrêter à 10).
FOR A = 10 TO 1 STEP -2 (la valeur du pas vaut -2).
FOR A = B/T TO B*T STEP X (les valeurs calculées sont également autorisées).

NEXT (N.) — Termine une boucle FOR . . . NEXT. Vérifie que la valeur d'index n'a pas dépassé la valeur finale, incrémente la valeur d'index de la valeur STEP et reprend l'exécution après FOR. Si l'index dépasse la valeur finale, le programme passe à l'instruction suivant NEXT.
Ex.: NEXT A (A est la variable d'index)

POP — Enlève l'information de retour stockée dans la pile et qui concerne la dernière instruction FOR ou GOSUB exécutée. Pratique pour sortir prématurément d'une boucle FOR—NEXT, ou pour sortir d'un sous-programme sans exécuter RETURN.
Ex.: POP: GOTO 10

IF. . .THEN — L'instruction qui suit THEN est exécutée lorsque la condition qui se trouve entre IF et THEN est vraie. Sinon, passe à la ligne suivante du programme.
Ex.: IF A ≠ C THEN GOTO 300
IF A + B THEN PRINT "A = B":
PRINT "FICHTRE!": LET A = 5:
GOTO 20
IF A THEN PRINT "A est différent de 0" (une expression différente de 0 es vraie)

TRAP (T.) — Lors d'une erreur, TRAP déroute l'exécution à la ligne spécifiée. TRAP reste valide jusqu'à ce qu'une erreur se présente ou bien jusqu'à la prochaine instruction TRAP.
PEEK (195) renvoie le numéro d'erreur.
Ex.: TRAP 30 (en cas d'erreur, aller à la ligne 30)
TRAP 40000 (les numéros de ligne plus élevés que 32767 mettent TRAP hors fonction).

STOP (STO.) — Arrête le programme. Imprime le numéro de ligne. Ne ferme pas les fichiers ni n'arrête le son. CONT. permet de relancer un programme.
Ex.: IF A=B THEN STOP

END — Arrête le programme, ferme tous les fichiers ouverts, arrête le son. Le programme peut être relancé avec CONT.

ENTREE/SORTIE

DESIGNATION DES PERIPHERIQUES

Chaque appareil de la gamme ATARI a une désignation qui lui est propre. L'unité de disquette A 810 et les ports série du module d'interface A 850™ (interface RS 232) doivent être numérotés de 1 à 4. Les unités de disquette doivent aussi recevoir un nom de fichier. Ces noms dovient être placés dans des chaînes de caractères ou dans des variables chaînes de caractères. En voici quelques exemples:

K: Clavier. Entrée seulement.
P: Imprimante. Sortie seulement.
C: Cassette. Entrée et Sortie.
S: Ecran (TV). Sortie seulement.
E: Edition à l'écran (clavier et écran combinés).
Entrée et Sortie.
R: Interface RS 232. Port série du module d'interface A 850. Entrée et Sortie.

D:NOM.EXT Exécute le fichier sur l'unité de disquette n° 1. NOM.EXT placé dans la disquette n° 1.
D2:NOM.EXT Même fichier NOM.EXT sur l'unité de disquette ° 2.

INSTRUCTIONS D'ENTREES/SORTIES

Les noms de fichiers mis sur disquette commencent par une lettre de l'alphabet et peuvent avoir une longueur allant jusqu'à 8 caractères. La désignation du fichier peut se terminer par une extension optionnelle (un point suivi de 1 à 3 caractères). Vous pouvez utiliser n'importe quelle combinaison de 3 lettres ou chiffres à votre gré. Voici quelques extensions utiles possibles:
.BAS = Sauvegarde des programmes BASIC (identique à SAVE).
.LST = Listage des programmes BASIC (identique à LIST).
.DAT = Fichier de données
.OBJ =Fichier en langage machine (code objet)
.TXT =Fichier de texte

OPEN (O.) — Ouvre un canal et lui associe un buffer en vue d'une opération d'entrée /sortie. Les valeurs possibles de canaux vont de 1 à 7. Utilisez les codes suivants:

TYPE	CODE	OPERATION
Lecture	4	Lecture seulement
Ecriture	8	Ecriture seulement
Ecriture & Lecture	12	Lecture et écriture
Ajout	9	Ajout en fin de fichier
Catalogue	6	Catalogue de la disquette considérée

Ex.: OPEN # 1, 4, 0, "K:" (Ouverture du canal n° 1 pour le clavier (lecture))
OPEN #2, 8, 0, "P:" (Ouverture du canal n° 2 pour l'imprimante (écriture))
OPEN # 1, 12, 0, "D:FICHIER.DAT" (Ouverture du canal n° 1 pour le fichier FICHIER DAT placé sur disquette (écriture et lecture))
OPEN # 1, 6, 0, "D:*.*" (Ouverture du canal n° 1 pour lire le catalogue de la disquette placée dans l'unité 1)

CLOSE (CL.) — Ferme un canal et après l'opération d'entrée ou de sortie et desalloue le IOCB. Peut être exécutée lorsqu'aucun canal n'a été ouvert. Les numéros IECO sont 1 7, comme dans OPEN.
Ex.: CLOSE 1 (Ferme le canal ouvert sur IOCB No. 1 et désalloue le IOCB.)

INPUT (I.) — Reçoit une ligne de caractères par un canal. La chaîne de caractères doit être terminée par un caractère "fin de ligne" (EOL).
Ex.: INPUT A (Entrée d'une valeur numérique (sous-entendue au clavier)).
INPUT A, B, C, (Entrée de trois valeurs numériques séparées par des virgules et placement dans les trois variables A, B, C)
INPUT A $ (Entrée d'une ligne de caractères. La placer dans la variable chaîne A $; A $ ne contient pas de caractère RETURN)
INPUT # 1; A $ (Entrée d'une ligne de caractères par le canal n° 1 et mise en place de ces caractères dans la nouvelle chaîne A$ and B)

PRINT (PR.) ou **(?)** — Envoie des données sur l'écran ou sur autre équipment.
Ex.: PRINT (envoie une ligne vierge).
PRINT "Le numéro est" ; A (Imprime texte et numéro à l'écran.)
PRINT "Le numéro est", A (La virgule entraîn l'impression de A dans une colonne séparée. POKE 201 avec largeur de colonne désirée.)
PRINT A $; (Le point-virgule empêche le RETURN D'être envoyé à la fin de la line.)
PRINT #1 ; A $ (Envoie A$ aux périphériques ouverts sur IOCB No.1.)

LPRINT (LP.) — Envoie des messages ou des résultats vers l'imprimante.
L'ouverture et la fermeture d'un canal ne sont pas nécessaires. Le point-virgule placé à la fin de l'instruction provoque toutefois un saut de ligne.
Ex.: LPRINT Provoque un saut de ligne.
LPRINT A$ (A$ seront imprimés)
LPRINT A$;B(A$ et B seront sur la même ligne.)
LPRINT A$,B (La virgule entraîn l'impression de B dans une colonne séparée. POKE 201 avec largeur de désirée.)

GET — Attend un octet en provenance du périphérique spécifié par le numéro de canal et le place dans la variable indiquée.
Ex.: GET # 1,A (Attend un octet du périphérique relié au canal n° 1 et le place dans la variable A).

PUT — Envoie un octet à destination du périphérique spécifié.
Ex.: PUT 1, A (Insére l'octet qui est dans A, au cnal ouvert sur IOCB 1.)

NOTE (NO.) — Utilisé avec l'unité de disquette pour déterminer la position de l'octet suivant à lire ou à écrire dans l'ensemble du buffer.
Ex.: NOTE 1, SEC, BYTE (Place le numéro de secteur dans SEC et le numéro de l'octet à l'intérieur du secteur dans BYTE. Se réfère au fichier ouvert sur IOCB 1.)

POINT (P.) — Utilisé pour informer DOS de l'endroit où le prochain octet doit être pris et lu ou doit être adressé.
Ex.: POINT # 1, SEC, BYTE Positionne le pointeur sur le secteur numéro SEC, avec un offset d'une valeur BYTE.

STATUS (ST.) — Demande l'état du périphérique choisi.
Ex.: STATUS #1, A (Place l'état du périphérique relié au canal numéro 1 dans la variable A).

INSTRUCTIONS DE TRAITEMENT

LET (LE.) — Assigne des valeurs aux variables numériques ou de chaîne.
Ex.: LET A = B (valeur de B assignée à A)
LET A $ = "BONJOUR"
A = B: A$= "BONJOUR" (LET peut être omis)

POKE — Place un nombre entre 0 et 255 dans une case mémoire numérotée entre 0 et 65535. Les valeurs décimales sont arrondies. PEEK permet de lire ce qui se trouve en mémoire.
Ex.: POKE 82, 0 (Met 0 dans la case mémoire 82)
A = PEEK (82) (Lit le contenu de la case mémoire 82 et place le résultat dans la variable A)

DIM — Réserve une zone mémoire pour une chaîne de caractères ou des tableaux numériques. Chaque caractère occupe un octet et chaque élément d'un tableau numérique occupe six octets.
Ex.: DIM A$ (10) (On réserve la place pour une chaîne de caractères comprenant au minimum 10 caractères)
DIM B (10) (On crée un tableau numérique de 10 éléments à une dimension)
DIM B (10, 10) (On crée un tableau numérique de 100 valeurs à deux dimensions)
DIM A$ (10), B(10) (Séparez les différentes variables par des virgules)

COM — Voir DIM.

CLR — Efface les dimensions de n'importe quel tableau, vide les variables-chaînes, et réduit les variables numériques à zéro (inverse de DIM).
Ex.: CLR

DATA (D.) — Présente une liste de valeurs numériques et/ou de caractères alphanumériques qui seront lus séquentiellement par l'instruction READ.
Ex.: DATA 12, 27, BONJOUR, ATARI 600

READ (REA.) — Lit séquentiellement les données placées dans les lignes de DATA et les assigne à des variables. Il doit y avoir correspondance entre le type de la variable et le type de DATA (numérique ou alphanumérique).
Ex.: READ A (A sera le numéro suivant sun la note dans les instructions du DATA)
READ A$ (Sert également pour les chaînes de caractères).
READ A, A$, B, BS (Sépares les éléments mulitples par des virgules).

RESTORE — Positionne le pointeur utilisé par l'instruction READ sur une ligne de DATA particulière.
Ex.: RESTORE (L'oclet suivant sera le prémières de DATA).
RESTORE (10) (L'oclet suivant sera le prémier élément des instructions prémières de DATA à la ligne 10).

REM (R.) ou ([ESPACE].) — Permet l'introduction de commentaires dans le corps du programme. Tous les caractères ou instructions placés sur une ligne après une instruction REM sont ignorés par le programme.

GRAPHIQUES

GRAPHICS (GR.) — Sélectionne le mode graphique. Le fait d'ajouter 16 au numéro du mode graphique sélectionne le plein écran. En ajoutant 32, on supprime l'effacement automatique de l'écran.
Ex.: GRAPHIQUES 8 (Graphiques Made 8 avec écran partage)
GRAPHIQUE 8 8+16 (Modes, plein écran)
GRAPHIQUES 8+32 (N'effacera pas l'écran)
GRAPHIQUES 8+16+32 (Une combinaison des deux options)

SETCOLOR (SE.) — Définit la teinte et la luminance pour un registre couleur particulier.
Ex.: SETCOLOR 1, 2, 4 (Attribue au registre couleur 1, 1a couleur 2 et la luminance 4. Les registres vont de 0 à 3; les teintes de 0 à 15; les luminances de 0 à 14, numéros pairs seulement).
(L'exception est modes GTIA, qui peuvent utiliser des nombres pairs et impairs jusqu'á 15.)

COLOR (C.) — Dans les modes graphiques 3 à 11, cette instruction sélectionne le registre couleur qui sera utilisé dans l'instruction PLOT.
Ex.: COLOR 2 (Sélectionne le deuxième registre couleur dans les modes 0-2, sélectionne la caractère ASCII, ayant une valeur de deux pour PLOT).

FORMATS D'ECRAN ET MODES GRAPHIQUES

Mode Graphique	Type	Nombre de Colonnes	Nombre de Lignes* Ecran Partagé	Nombre de Lignes Plein Ecran	Nombre de Couleurs	Espace Mémoire Nécessaire Ecran Partagé	Espace Mémoire Nécessaire Plein Ecran
0	TEXTE	40	—	24	1,5	—	992
1	TEXTE	20	20	24	5	674	672
2	TEXTE	20	10	12	5	424	420
3	GRAPHIQUE	40	20	24	4	434	432
4	GRAPHIQUE	80	40	48	2	694	696
5	GRAPHIQUE	80	40	48	4	1174	1176
6	GRAPHIQUE	160	80	96	2	2174	2184
7	GRAPHIQUE	160	80	96	4	4190	4200
8	GRAPHIQUE	320	160	192	1-1/2	8112	8138
9	GRAPHIQUE	80	—	192	1	—	8138
10	GRAPHIQUE	80	—	192	9	—	8138
11	GRAPHIQUE	80	—	192	16	—	8138
12	GRAPHIQUE	40	20	24	5	1154	1152
13	GRAPHIQUE	40	10	12	5	664	660
14	GRAPHIQUE	160	160	192	2	4270	4296
15	GRAPHIQUE	160	160	192	4	8112	8138

PLOT (PL.) — Affiche un point ou un caractère à l'écran aux coordonnées X-Y spécifiées.
Ex.: PLOT 3, 5 (troisième colonne, cinquième ligne)

POSITION (POS.) — Positionne le curseur sur l'écran à un emplacement de coordonnées X et Y. Mais ne trace rien.
Ex.: POSITION X, Y

LOCATE (LOC.) — Permet de relire dans la mémoire-écran le contenu d'une case spécifiée. Dans les modes 0 à 2, on relit un caractère. Dans les modes 3 à 11, on lit le numéro de couleur.
Ex.: LOCATE X Y D (Avance à X, Y et met des données dans D).

DRAWTO (DR.) — Tire une ligne entre la dernière position du curseur et le point de coordonnées X et Y. Le curseur s'arrête aux nouveaux coordonnées. (Tire une ligne au point X, Y de la position actuelle du curseur).
Ex.: DRAWTO X-Y

VALEURS PAR DEFAUT DES COULEURS A LA MISE SOUS TENSION

Numéro du Registre Couleur	Couleur Par Défaut	Luminance Par Défaut	Resultat
0	2	8	ORANGE
1	12	10	VERT
2	9	4	BLEU FONCE
3	4	6	ROSE OU ROUGE
4	0	0	NOIR

Il y a "DEFAUT" si aucune indication de REGISTRE DE COULEUR n'est utilisée.

Remarque: Les couleurs peuvent varier selon le type de moniteur de télévision et l'état et réglage de ce dernier.

TABLEAU DE CORRESPONDANCE ENTRE LES COULEURS ET LES VALEURS NUMERIQUES (INSTRUCTION SET COLOR)

Couleurs	Werte Argument Numéro 2
GRIS	0
ORANGE CLAIR (OR)	1
ORANGE	2
ORANGE-ROUGE	3
ROSE	4
VIOLET	5
BLEU-VIOLET	6
BLEU	7
BLEU	8
BLEU CLAIR	9
TURQUOISE	10
BLEU-VERT	11
VERT	12
VERT-JAUNE	13
VERT-ORANGE	14
ORANGE CLAIR	15

Remarque: Les couleurs peuvent varier selon le type et le réglage du téléviseur ou du moniteur utilisé.

FONCTIONS

Une fonction accepte un ou plusieurs arguments pour donner finalement une nouvelle valeur. L'ordinateur ATARI possède des fonctions arithmétiques, des fonctions trigonométriques, des fonctions chaînes de caractères et quelques fonctions spéciales.

FONCTIONS ARITHMETIQUES

ABS — Donne la valeur absolue (sans signe) d'un nombre.
Ex.: A = ABS (B)

CLOG — Donne le logarithme base 10.
Ex.: A = CLOG (B)

EXP — Donne la valeur e (2,718 approximativement) élevée à la puissance spécifiée. Dans certains cas, la précision d'EXP ne s'étend que sur 6 digits significatifs. Inverse de LOG.
Ex.: A = EXP (B)

INT — Donne la partie entière de la valeur.
Ex.: A = INT (B)

LOG — Donne le logarithme naturel d'une valeur. Inverse de EXP.

RND — Tire un nombre aléatoire entre 0 et 1. Ne donne jamais 1.
Ex.: A = RND (0) (A = un nombre égal à 0 ou plus petit que 1)
A = RND (0) x 8 (A = un nombre égal où supérieur à 0, et plus petit que 8)

SGN — Donne — 1 si la valeur est négative, 0 si la valeur est nulle et 1 si la valeur est positive.
Ex.: A = SGN (A)

SQR — Donne la racine carrée positive d'une valeur positive.
Ex.: A = SQR (B)

FONCTIONS TRIGONOMETRIQUES

ATN — Donne l'arctangente d'une valeur, en radians et degrés.
Ex.: A = ATN (B)

SIN — Donne le sinus de la valeur.
Ex.: A = SIN (B)

COS — Donne le cosinus d'une valeur.
Ex.: A = COS (B)

DEG — Toutes les fonctions trigonométriques seront exprimées en degrés.
Ex.: DEG

RAD — Toutes les fonctions trigonométriques seront exprimées en radians. A la mise sous tension, l'ordinateur travaille en radians.

FONCTIONS SPECIALES

ADR — Donne l'adresse mémoire décimale du début d'une chaîne.
Ex.: A = ADR (B$)
A = ADR ("CETTE CHAINE")

FRE (O) — Donne le nombre d'octets de mémoire vive utilisateur qui restent. (0) est une variable "fictive" exigée.
Ex.: A = FRE (0)
PRINT FRE (0) (Ce qui vous permettra de savoir combien il reste de mémoire libre).

PEEK — Donne le nombre stocké à un endroit spécifique de la mémoire. L'adresse est un nombre compris entre 0 et 65535. Le nombre lu sera compris entre 0 et 255.
Ex.: A = PEEK (B)

USR — Appel des sous-programmes en langage machine à partir de BASIC. USR(ADDR, P1, P2, . . . , PN), par exemple, pousse les arguments (P1 à PN) dans la pile dans l'ordre inverse. Le dernier argument "PN" est poussé dans la pile avant le premier, "P1".
Le nombre d'arguments — représenté par un seul octet — est alors poussé dans la pile. Si la fonction ne spécifie pas d'arguments, c'est zéro qui est poussé dans la pile.
Le sous-programme en langage machine à l'adresse (ADDR) est alors appelé. Si le programme en langage machine doit donner une valeur en BASIC, les octets de poids fort et de poids faible doivent être stockés aux adresses de mémoire $D4 et $D5 respectivement.
Le programme doit éliminer le nombre d'arguments et les arguments avant de revenir à BASIC, sinon le système sera complètement déréglé.
Ex.: A = USR (B, C, D)
(Le programme en B sera appelé, et les paramètres en C et D passeront au sous-programme par l'intermédiaire de la pile).

FONCTIONS DE CHAINE

ASC — Donne le numéro de code ATASCII pour le premier caractère d'une chaîne.
Ex.: A = ASC ("A") (A vaudra 65)
A = ASC (B$) (On peut utiliser des variables de chaînes)

CHR$ — Donne le caractère représenté par le numéro de code ATASCII qui a été spécifié. Réciproque de ASC.
Ex.: A$ = CHR$ (65) (A$ sera "A")

LEN — Donne un nombre qui représente la longueur d'une variable de chaîne.
Ex.: A = LEN (A$)

STR$ — Donne une chaîne représentant une valeur spécifiée (traduit un nombre en une chaîne).
Ex.: A$ = STR$ (65) (A$ sera égal à "65", qui est une chaîne)

VAL — Donne un nombre représentant une chaîne spécifique (traduit une chaîne en un nombre)
Ex.: A = VAL ("100") (A sera égal au nombre 100)

MANIPULATION DE CHAINES

Le BASIC ATARI n'emploie pas un format en tableau de chaîne pour la manipulation des chaînes. Une puissante commande d'extraction de chaîne suffit pour toutes les applications.
Exemples:
CHAÎNE EXTRAITE
50 A$ = "DAVEMARKGRETCHEN"
60 B$ = A$(9, 16) (B$ est "GRETCHEN")

CONCATENATION
50 A$ = "HI"
60 B$ = "FRED"
70 A$ (LEN (A$) + 1) = B$ (A$ est "HI FRED")

RECHERCHE D'UNE CHAINE
50 FOR Z = 1 TO LEN (A$)
60 IF A$ (Z, Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z

FONCTIONS DE COMMANDE DE JEU

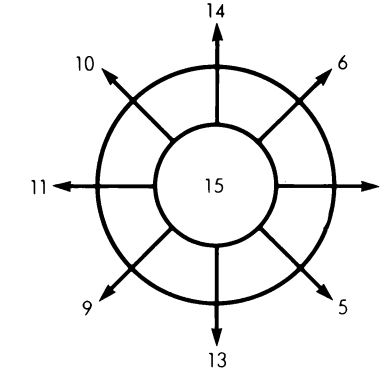
PADDLE — Donne la position du bouton d'une commande à molette spécifique. Les commandes sont numérotées de 0 à 3, de l'avant vers l'arrière. Le nombre donné se situe entre 1 et 228, augmentant au fur et à mesure que la molette est tournée vers la gauche (sens inverse des aiguilles d'une montre).
Ex.: A = PADDLE (0)

PTRIG — Le résultat est 0 si l'on presse le bouton rouge de la commande à molette correspondante, 1 dans le cas contraire. Les commandes sont numérotées de 1 à 3.
Ex.: A = PTRIG (0)

STICK — Donne l'état d'une commande à levier spécifique. Les commandes à levier sont numérotées 0 et 1, de l'avant vers l'arrière. Pour les détails, voir le diagramme.
Ex.: A = STICK (0)

STRIG — Donne un 0 si l'on presse le bouton rouge de la commande à levier correspondante, et 1 dans le cas contraire. Les commandes à levier sont numérotées 0 et I.
Ex.: A = STRIG (0)
Commande 1 = STICK (0)
Commande 2 = STICK (1)

Le diagramme qui suit donne les valeurs qui sont renvoyées lorsque la commande à levier est déplacée dans une direction quelconque.



TOUCHES DE FONCTIONS SPECIALES

ESC — A pour effet d'afficher la touche suivante sous forme de caractère international ou de caractère graphique.

BREAK — Provoque l'arrêt d'un programme BASIC.

RESET — Arrête un programme en cours d'exécution, remet l'écran en mode graphique 0, efface l'écran et met fin à vos fichiers sans les fermer. N'efface pas votre programme.

SET-CLR-TAB — Déplace le curseur jusqu'à la prochaine position de tabulation prédéterminée.

SHIFT SEF-CLR-TAB — Elimine une tabulation.

CTRL 1 — Fige ou libère le défilement de l'affichage à l'écran.

CTRL 2 — Produit un son.

CTRL 3 — Indique la fin d'un fichier.

MISE AU POINT

SHIFT INSERT — Insère une ligne.

CTRL INSERT — Insère un caractère

DELETE BACK S — Supprime le caractère à gauche du curseur et fait reculer le curseur.

SHIFT DELETE BACK S — Supprime une ligne.

CTRL DELETE BACK S — Supprime le caractère qui se trouve au curseur et déplace le reste de la ligne pour remplir l'espace vide.

SHIFT CLEAR (ou) **CTRL CLEAR** — Efface l'écran.

CTRL ↑ Déplace le curseur vers le haut.

CTRL ↓ Déplace le curseur vers le bas.

CTRL ← Déplace le curseur vers la gauche.

CTRL → Déplace le curseur vers la droite.

CODE DES ERREURS

NUMERO DE CODE D'ERREUR	MESSAGES DES CODES
2	Mémoire insuffisante.
3	Valeur erronée.
4	Trop de variables.
5	Erreur de longueur pour une chaîne de caractères.
6	Dépassement de données.
7	Nombre plus grand que 32767.
8	Erreur dans une instruction INPUT.
9	Erreur de dimension (DIM) de tableaux ou de chaînes de caractères.
10	Dépassement de pile.
11	Dépassement de capacité en virgule flottante.
12	Ligne inexistante.
13	Instruction NEXT sans FOR.
14	Ligne trop longue.
15	Suppression de l'instruction GOSUB ou FOR.
16	Erreur sur une instruction RETURN.
17	Erreur de mémoire.
18	Caractère non valide.
19	Chargement d'un programme (LOAD) trop long.
20	Erreur du numéro de périphérique.
21	Erreur de chargement. (LOAD)

REMARQUE: Vour trouverez ci-dessous les erreurs d'ENTREE/SORTIE découlant de l'utilisation des unités de disquettes, imprimantes ou autre équipement accessoire. De plus amples informations sont fournies avec le matériel ("hardware") auxiliaire.

128	BREAK pendant une opération d'entrée /sortie.
129	IOCB déjà ouvert.
130	Périphérique inexistant.
131	Opération de lecture sur un périphérique ouvert en écriture seulement. (IOCB)
132	Commande invalide.
133	Périphérique ou fichier non ouvert.
134	Mauvais numéro d'IOCB.
135	Tentative d'écriture vers un périphérique ouvert en lecture seule. (IOCB)
136	Fin de fichier. (EOF)
137	Enregistrement incomplet.
138	Pas de réponse du péri phérique dans le temps normal.
139	Le périphérique ne répond pas correctement. (NAK)
140	Erreur sur le bus série en entrée.
141	Le curseur dépasse la gamme autorisée.
142	Erreur de transmission dans le bus série.
143	Erreur sur le bus série en sortie.
144	Erreur d'opération sur le périphérique.
145	Erreur de vérification après écriture sur disque.
146	Fonction non implémentée.
147	Mémoire insuffisante.
160	Erreur sur le numéro de disque.
161	Trop de fichiers ouverts (OPEN) simultanément.
162	Disquette pleine.
163	Erreur système en entrée/ sortie non corrigeable.
164	Erreur de chaînage entre les sectreurs du disque.
165	Erreur de longueur pour l'instruction POINT.
167	Fichier verrouillée.
168	Commande invalide.
169	Sommaire de disquette plein.
170	Fichier non trouvé.
171	Instruction POINT invalide.
172	Ajout interdit
173	Mauvais Format

Note: Voir page 11 (ERROR CODES) dans la version anglaise comment le message codé d'erreurs apparaitra sur l'écran de votre téléviseur.

HANDLEIDING ATARI BASIC TM

VOOR ERVAREN PROGRAMMEURS

Het leren van de programmeertaal BASIC is gelijk aan het leren van iedere andere taal — het kost een beetje tijd en moeite, maar deze moeite wordt met succes beloond. Deze handleiding verschaft informatie over ATARI BASIC — een populaire en krachtige taal —aan degenen, die reeds bekend zijn met de programmeertaal BASIC.

Deze handleiding is alleen bedoeld om als naslagwerk te worden gebruikt. Er worden geen veelomvattende programmeervoorbeelden behandeld of leerinformatie voor de beginner gegeven. Zowel beginners als ervaren programmeurs worden voor meer informatie verwezen naar de volgende boeken: "ATARI BASIC" van Albrecht, Finkel en Brown, "ATARI BASIC REFERENCE MANUAL" en "ATARI BASIC VAN BINNENUIT BEKEKEN" van Bill Carris.

INDEX

REKENKUNDIGE BEWERKINGEN

Bewerkingen die tussen het binnenste paar haakjes worden aangegeven, worden het eerst uitgevoerd en dan op het volgende niveau overgedragen. Wanneer paren van haakjes in een ander paar ingesloten zijn noemt men ze "genesteld" ("nested"). Handelingen die zich op hetzelfde genestelde niveau bevinden moeten in de volgende volgorde worden uitgevoerd:

HOOGSTE PRIORITEIT/ LAAGSTE PRIORITEIT

<, >, =, <=, >=, <>
Vergelijkende bewerkingstekens in alfanumerieke gegevens (strings) hebben dezelfde prioriteit en moeten van links naar rechts worden uitgevoerd.

-
Monadisch minteken (duidt negatieve getallen aan).

^
Machtsverheffen.

***, /**
Vermenigvuldiging en deling hebben dezelfde prioriteit en moeten van links naar rechts worden uitgevoerd.

+, -
Optelling en aftrekking hebben dezelfde prioriteit en moeten van links naar rechts worden uitgevoerd.

<, >, =, <=, >=, <>
Vergelijkende bewerkingen voor numerieke gegevens (getallen) hebben dezelfde prioriteit en worden van links naar rechts uitgevoerd.

NOT
logische ontkenning

AND
logische AND

OR
logische OR

(Toegestane afkortingen tussen haakjes).
De ATARI huiscomputers maken geen onderscheid tussen commandos en opdrachten. De volgende woorden kunnen als programma-opdracht (statement) of als direct commando gebruikt worden door ze in te typen en de RETURN-toets in te drukken. Deze woorden mogen niet voor het benoemen van variabelen gebruikt worden.

SYSTEEM INSTRUKTIES

BYE (B.) — Treedt uit de BASIC- in de SELF TEST -mode.

DOS — Roept het DOS-menu op (alleen te gebruiken bij een DISK DRIVE. DOS = Disk Operating System, het besturingssysteem van de disk drive.)

CSAVE (CS.) — Staat een programma uit het geheugen op een cassette op.

CLOAD — Laadt een programma vanaf een cassette in het geheugen.

SAVE (S.) — Stuurt een BASIC-programma naar een randapparaat (bijv. disk drive) voor opslag.
Bijv.: SAVE "D:MIJNBEST.BAS".

LOAD (LO.) — Laadt een programma in het geheugen vanuit een gegevensinvoerapparaat (bijv. disk drive).
Bijv.: LOAD"D:MIJNBEST.BAS".

LIST (L.) — Schrijft een programma op het beeldscherm of stuurt het naar een randapparaat.
Bijv.: LIST: (Schrijft het gehele programma op het scherm)
LIST 10 (Schrijft regel 10 op het beeldscherm)
LIST 10, 20 (Schrijft regel 10 tot en met 20 op het beeldscherm)
LIST "P:" (Schrijft (print) het programma op een printer)
LIST "P:", 10 20 (Schrijft de regels 10 tot en met 20 op een printer)
LIST "D:MIJNBEST.LST" (Schrijft het programma weg naar diskette bestand — terughalen met ENTER)
LIST "D:MIJNBEST.LST", 10, 20 (Schrijft de regels 10 tot en met 20 weg naar een diskette bestand — terughalen met ENTER)
LIST "C:" (Schrijft het gehele programma weg naar cassette — terughalen met ENTER "C")

ENTER (E.) — Laadt een programma dat oorspronkelijk is opgeslagen met een "LIST" commando en voegt deze programmaregels toe aan een eventueel in het geheugen aanwezig programma.
Bijv.: ENTER "C:"
ENTER "D : MIJNBEST.LST"
Over gewone regelnummers in een programma, die reeds in de computer aanwezig zijn, wordt heengeschreven.

NEW — Wist het programma uit het geheugen.

RUN — Begint de uitvoering van een programma in BASIC. Het programma kan daarbij in het geheugen opgeslagen zijn of van een diskette of cassette worden geladen. (Maakt variabelen nul en wist de toegekende dimensies van reeksen en strings (zie p.p.10).
Bijv.: RUN (voert een in het geheugen opgeslagen programma uit).
RUN "D: MIJNBEST.BAS" (Laadt een programma van de diskette en voert het uit).

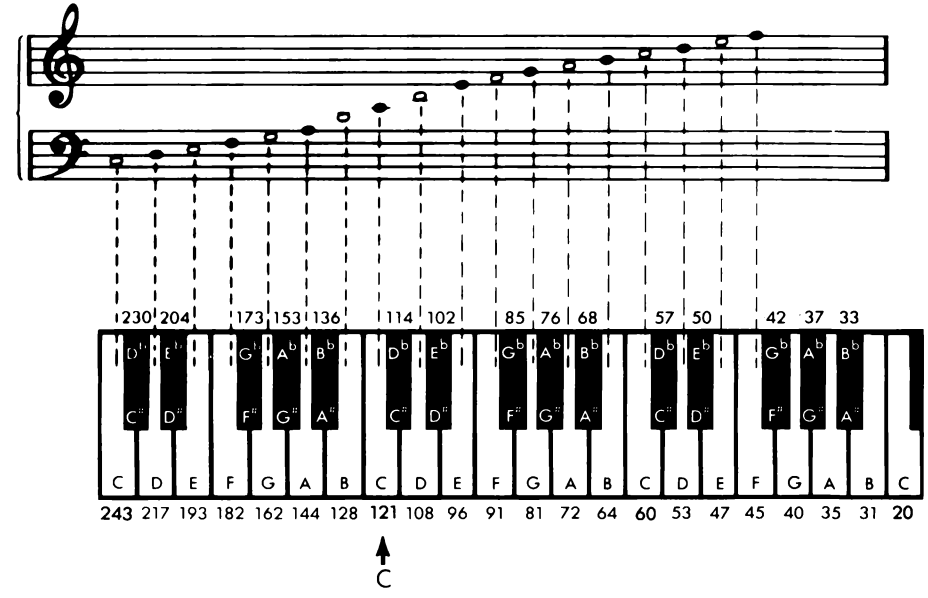
CONT — Zet de uitvoering van het programma voort, nadat de BREAK-toets ingedrukt was of het programma een STOP of END uitgevoerd heeft. Als er op dezelfde regel aanvullende programma-aanwijzingen staan worden deze niet uitgevoerd. Het programma gaat verder met de volgende genummerde regel.

GELUID

SOUND (SO.) — Activeert een van de vier kanalen om een geluids-signaal door de TV-luidspreker te laten klinken. Het geluid blijft hoorbaar totdat een andere SOUND-opdracht op hetzelfde kanaal het vorige commando overschrijft, of een END-, RUN- of NEW-commando uitgevoerd wordt.
De kanalen zijn onafhankelijk van elkaar programmeerbaar en kunnen alle vier tegelijkertijd klinken.
De SOUND (SO.) opdracht moet door vier waarden gevolgd worden (cijfers, variabelen of uitdrukkingen).

Bijv.: SOUND A, B, C, D waarbij:
A = Kanaalnummer (0-3)
B = Toonwaarde (0-255). Hoe groter deze waarde, des te lager is de frequentie. Frequentie = 31960/Toonwaarde + 1). Zie tabel voor muzikale equivalenten.
C = Vervorming (0-14, alleen even getallen). 10 en 4 zijn "zuivere" tonen. Bij de overige getallen worden andere geluiden geproduceerd.
D = Volume (0-15). Hoe groter de factor, des te groter het volume. 0 betekent uit. Wanneer het totale volume van alle vier de kanalen het getal 32 overschrijdt, kan de luidspreker gaan "brommen".

BETREKKING VAN HET KLAVIER OP DE TOONSCHAAL



PROGRAMMA INSTRUKTIES

GOTO (G.) — Programma uitvoering verolgt op het gespecificeerde regelnummer (betekent "ga naar . . .")
Bijv.: GOTO 30 (Ga naar regel 30 — vervolg het programma op regel 30)
Bijv.: GOTO A + 10 (Ga naar regelnummerwaarde A plus 10 — dit is toegestaan, maar bij een foutmelding moeilijk op te sporen).

ON . . . GOTO — De uitvoering van het programma wordt vervolgd op het regelnummer, dat wordt aangeduid door een uitdrukking.
Bijv.: 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.) — (Betekent "ga naar subroutine regel . . .") De uitvoering van het programma wordt op het aangegeven regelnummer vervolgd. De eerste RETURN-instruktie laat de uitvoering teruggaan naar de instruktie, die onmiddellijk op deze GOSUB volgt.
Bijv.: GOSUB 30 (Ga naar de subroutine op regel 30).
GOSUB A+10 (Toegestaan, echter bij een foutmelding moeilijk op te sporen).

ON . . . GOSUB — De uitvoering van het programma wordt vervolgd op het regelnummer dat wordt aangeduid door een uitdrukking. Een RETURN-aanwijzing laat de uitvoering teruggaan naar een instruktie die onmiddellijk op deze ON . . . GOSUB-instruktie volgt.
Bijv.: 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).
Waarschuwing: Als de waarde van de expressie minder is dan 1 of groter dan het aantal regelnummers, dan zijn de resultaten onvoorzienbaar.

RETURN (RET.) — Stopt een subroutine en laat de uitvoering teruggaan naar de instruktie, die onmiddellijk volgt op de laatst uitgevoerde GOSUB-instruktie.
Bijv.: RETURN.

FOR (F.) — Deze instruktie bepaalt de begin-en eindwaarden van een indexvariabele, als ook de waarde waarmee deze elke keer moet worden verhoogd wanneer een FOR . . . NEXT-lus uitgevoerd wordt. De verhoogde waarde is 1, behalve wanneer door een STEP (stapgrootte)-instruktie anders gespecificeerd wordt.
Een NEXT-instruktie veroorzaakt dat de instrukties, die tussen de FOR en NEXT staan, herhaald worden.
Bijv.: FOR A = 1 TO 10 (A zal bij 1 beginnen, telkens met 1 verhoogd worden en bij 10 ophouden).
FOR A = 10 TO 1 STEP -2 (A begint bij 10 en wordt steeds verminderd met een stapgrootte -2).
FOR A = B/T TO B*T STEP X (Ook berekende waarden zijn geldig).

NEXT (N.) — Beëindigt een FOR . . . NEXT-lus. Controleert of de indexwaarde niet groter dan de eindwaarde geworden is, verhoogt de indexwaarde met de STEP-waarde en vervolgt de uitvoering van het programma met de aanwijzing na de FOR-instruktie. In het geval dat de indexwaarde de eindwaarde overschrijdt, wordt naar de instruktie die op NEXT volgt gegaan.
Bijv.: NEXT A (A is de indexvariabele).

POP — Verwijdert de terugkeer informatie die is opgeslagen over de laatst uitgevoerde FOR- of GOSUB-instruktie. Deze funktie is nuttig wanneer, bijvoorbeeld, voortijdig een FOR-NEXT-lus verlaten moet worden, of waneer uit een subroutine moet worden gesprongen zonder dat een RETURN is uitgevoerd.
Bijv.: POP: GOTO 10

IF . . . THEN — De opdracht na THEN wordt alleen uitgevoerd, als de voorwaarde tussen IF en THEN waar is. Anders wordt met de volgende programmaregel verder gegaan.
Bijv.: IF A ≠ C THEN GOTO 300
IF A = B THEN PRINT "A = B";
PRINT "HIER ZIJN WE DAN": LET A = 5:GOTO 20
IF A THEN PRINT "A IS NIET NUL"
(De uitdrukking "A is niet gelijk aan nul" is waar).

TRAP (T.) — Bij een fout stuurt TRAP het programma naar de gespecificeerde regel. TRAP blijft actief tot zich een fout voordoet, respectievelijk tot de volgende TRAP-instruktie. PEEK (195) geeft het nummer van de fout aan.
PEEK 187* 256+PEEK (186) geeft het regelnummer weer.
Bijv.: TRAP 30 (ga bij een fout naar regel 30)
TRAP 40000 (Nummers van de regels, die groter zijn dan 32767, schakelen TRAP uit).

STOP — Stopt het programma en toont het regelnummer waar gestopt is. Sluit de "bestand" echter niet (zie blz. 8) en zet ook het geluid niet uit. Een programma kan opnieuw worden begonnen met CONT.
Bijv.: IF A = B THEN STOP

END — Stopt het programma, sluit alle geopende bestanden en zet het geluid uit.

(INVOER UITVOER)

BENAMINGEN VAN DE APPARATUUR

Elk ATARI apparaat heeft een unieke benaming. Zowel diskette-stations als het ATARI 850™ Interface module (RS232-handler) verlangen een apparaatnummer (1-4). Diskette-stations verlangen ook een bestandsnaam moeten tussen aanhalingstekens staan of in een string**-functie zijn opgenomen.

Hier volgen enige voorbeelden:
K: Toetsenbord (Keyboard). Alleen voor gegevensinvoer (INPUT).
P: Printer. Alleen voor gegevensuitvoer (OUTPUT).
C: Cassette. Input en output.
S: Beeldscherm. Alleen output.
E: Scherm en toetsenbord (Editor; toetsenbord met beeldscherm gecombineerd). Input en output.
R: RS232-poort (ATARI 850 Interface module). Input en Output.
D:FILENAME.EXT "Bestandsnaam.EXT" Diskettestation 1.
D2:FILENAME.EXT Hetzelfde bestand op diskettestation 2.

**String = (betekenis "rijgnoer") een rij "aan elkaar geregen tekens" voorgesteld door een variabele naam, gevolgd door een $-(string)-teken. Zie ook DIM.

Namen van diskette bestandenbeginnen met een letter en kunnen tot 8 tekens lang zijn. De bestandsnaam kan met een facultatieve toevoeging (EXTension) eindigen. (Een punt, gevolgd door 1 tot 3 tekens). Men kan naar believen 3 letters of cijfers gebruiken. Sommige praktische verlengingen zijn:
.BAS = Bewaart BASIC programma's (juist als SAVE).
.LST = Noteert BASIC programma's (juist als LIST).
.DAT = Algemene gegevensbestanden.
.OBJ = Machinetaal bestanden ("OBJect bestanden")
.TXT = Tekst-bestanden

I/O — INSTRUKTIES

OPEN (O.) — Bereid een apparaat op input of output voor. De IOCB *— nummers zijn 1 t/m 7. Gebruikt de volgende codes:

BEWERKING	CODE	BETEKENIS
Input	4	alleen lezen
Output	8	alleen schrijven
Update	12	lezen en schrijven
Append	9	voeg toe aan het eind van het best.
Directory	6	alleen diskette inhoudsopgave (directory)

Bijv.: OPEN #1, 4, 0, "K:" Opent het toetsenbord voor input door IOCB # 1 (bewerking 4)
OPEN # 2, 8, 0, "P:" Opent de printer voor output door IOCB #2 (bewerking 8)
OPEN # 1, 12, 0, "D:MYFILE.DAT" Opent het diskettebestand voor "update"-functie door IOCB #1 (bewerking 12)
OPEN # # 1, 6, 0, "D:x.x" Opent het diskettestation 1 voor inhoudsopgave door IOCB # 1).

* IOCB is de afkorting van In/Out Control Block; inwendige poort-nummers waardoor de informatie wordt verstuurd.

CLOSE (CL.) — Sluit het apparaat af na een input- of outputhandeling en geeft het IOCB-nummer vrij. Mag ook gebruikt worden wanneer geen apparaat werd geopend. De IOCB-nummbers zijn 1 t/m 7 zoals bij OPEN.
Bijv.: Close #1 (Het bij IOCB #1 geopende bestand sluiten en IOCB vrijgeven)

INPUT (I.) — Haalt één regel tekens uit een apparaat. De regel moet door een RETURN-teken beëindigd worden.
Bijv.: INPUT A (Vraag een getal en zet het in A).
INPUT A, B, C (Vraag 3 getallen, gescheiden door komma's en zet ze in A, B en C)
INPUT A$ (Vraag een rij tekens en zet ze in A$; A$ bevat geen RETURN-teken)
INPUT 1, A$, B (Haal een rij tekens uit een door het IOCB#1 geopende apparaat en zet hem in A$ en B).

PRINT (PR.) of **(?)** — Betekent "afdrukken"; stuurt gegevens naar het beeldscherm of een ander apparaat.
Bijv.: PRINT (Er verschijnt een lege regel).
PRINT "Het getal is"; A (Tekst en getal worden op het beeldscherm achter elkaar afgedrukt).
PRINT "Het getal is", A (De komma zorgt ervoor, dat A in een aparte kolom wordt afgedrukt. POKE 201 met de gewenste kolombreedte.)
PRINT A$; (De punt-komma verhindert dat RETURN aan het eind van de regel meegezonden wordt. De eerstvolgende print-instruktie wordt dan ook op dezelfde regel afgedrukt.
PRINT #1, A$ (Schrijft A$ weg naar het apparaat dat is geopend op #1.

LPRINT (LP.) — Drukt gegevens af op de printer. Er is geen OPEN- of CLOSE-functie nodig. Een punt-komma aan het eind van de regel verhindert niet dat RETURN wordt meegezonden.
Bijv.: LPRINT (Zendt een lege regel naar de printer)
LPRINT A$ (A$ wordt afgedrukt)
LPRINT A$;B (A$ en B zullen op dezelfde regel worden afgedrukt).
LPRINT A$,B (De komma zorgt ervoor, dat B in een aparte kolom wordt afgedrukt. POKE 201 met de gewenste kolombreedte.)

GET — Neemt één enkele byte (teken) uit het gespecificeerde apparaat en zet het in de aangegeven variabele.

PUT — Stuurt een enkele byte in een variabele naar een gespecificeerd apparaat.
Bijv.: PUT #1,A (Stuurt het byte in A naar het met IOCB #1 geopende apparaat).

NOTE (NO.) — Wordt met diskette gebruikt ter bepaling van de plaats van het volgende te lezen of te schrijven byte.
Bijv.: NOTE #1, SEC, BYTE (Zet het sectornummer in SEC en het bytenummer in BYTE; verwijst daarbij naar het door IOCB #1 geopende bestand).

POINT (P.) — Wordt gebruikt om de DOS te vertellen, waar het volgende te lezen of te schrijven byte zich bevindt.
Bijv.: POINT #1, SEC, BYTE (Verwijst DOS haar sectornummer SEC, en BYTE-nummer BYTE).

STATUS (ST.) — Leest de statuswaarde van een gespecificeerd apparaat. De zo gehaalde statuswaarde wordt verklaard in de tabel met foutmeldingen.
Bijv.: STATUS #1,A (Zet de statuswaarde van het door IOCB #1 geopende apparaat in A).

BEWERKINGSINSTRUKTIES

LET — Kent waarden aan numerieke of string-variabelen toe.
Bijv.: LET A=B (De waarde van B wordt aan A toegekend)
LET A$="HELLO"
A=B; A$="HELLO" (Gebruik van het woord LET is niet verplicht).

POKE — Zet een getal tussen 0 en 255 in een bepaalde geheugenplaats tussen 0 en 65535. Gebroken getallen worden afgerond. Gebruik PEEK om een geheugenplaats te lezen.
Bijv.: POKE 82,0 (Zet waarde 0 in de geheugenplaats 82)
A=PEEK (82) (Leest de inhoud van de geheugenplaats 82 en kent deze waarde toe aan variabele A. Haakjes zijn verplicht bij PEEK).

DIM — Dimensionering. Reserveert geheugenruimte voor strings en numerieke reeksen. Elk stringteken, neemt één byte; elk element in een numerieke reeks neemt zes bytes.
Bijv.: DIM A$(10) (Een string met 10 variabelen)
DIM B(10) (Een numerieke reeks; B bevat elementen 0 t/m 10)
DIM B(10,10) (Een tweedimensionale reeks)
DIM A$(10),B(10) (Verschillende variabelen worden door komma's van elkaar gescheiden).

COM — Gelijk aan DIM.

CLR — Verwijdert de dimensies van elke opstelling en brengt reeksen en variabele getalwaarden terug tot nul.
Bijv.: CLR

DATA (D.) — Instruktie die een lijst van cijfers en/of letters bevat, die gebruikt kunnen worden bij de hieronder vermelde READ-instruktie.
Bijv.: DATA 1,2,3,4,A,B,C,D (Een lijst met informatie die moet worden gelezen door een "READ"-instruktie).

READ — Leest het volgende onderdeel uit een DATA-regel en kent het toe aan een variabele. Als een DATA regel is gebruikt haalt READ data van de volgende DATA-regel in het programma.
Bijv.: READ A (A zal het volgende getal op de lijst in de DATA-regel zijn).
READ A$ (Geldt ook voor strings)
READ A,A$,B,B$ (Houdt meervoudige variabelen met komma's uit elkaar).

RESTORE (RES.) — Verwijst READ naar een DATA-regel.
Bijv.: RESTORE (Het volgende data-punt wordt de eerste post van de eerste DATA-lijst.)
RESTORE 10 (Het volgende data-punt wordt de eerste post van de eerste Data-lijst.)

REM (R.) of ([SPACE].) — Laat verklarende opmerkingen in een programma toe.

Alles wat vanaf REM tot aan het eind van de regel volgt, wordt door BASIC genegeerd.
Bijv.: REM Dit is een opmerking!

BEELDSCHERM INSTRUKTIES

GRAPHICS (GR.) — Kiest een grafische modus of beeldschermindeling. Modus + 16 kiest een ongedeeld beeldscherm (zonder tekstvenster). Modus + 32 voorkomt wissen van het beeldscherm.
Bijv.: GRAPHICS 8 (Modus voor Graphics 8 met tekstvenster)
GRAPHICS 8+16 (Modus 8, gehele scherm beschikbaar voor Graphics)
GRAPHICS 8+32 (Het beeldscherm wordt niet gewist)
GRAPHICS 8+16+32 (Beide mogelijkheden gecombineerd)

SETCOLOR (SE.) — Bepaalt de kleurnuance en helderheid van het gekozen kleurregister. Het registernummer is niet hetzelfde als bij de COLOR-opdracht.
Bijv.: SETCOLOR 1,2,4 (Stelt register 1 in op kleur 2 in helderheid 4)
Register: 0-4
Kleur: 0-15
Helderheid: 0-14 (alleen even getallen; behalve GTIA modi, die zowel even als oneven nummers tot 15 kunnen gebruiken).

COLOR (C.) — In de grafische modes (3-11) wordt een kleurregister voor het gebruik van PLOT gekozen. Het register hier is niet gelijk aan het register in SETCOLOR.
Bijv.: COLOR 2 (Kiest het kleurregister 2. In de modi 0-2 wordt een ASCII-teken gekozen, waarvan de waarde 2 is voor PLOT.)

TABEL MET BEELDSCHERMMODI EN -INDELINGEN

BEELDSCHERMINDELING

Graphics Modus	Type	Aantal kolommen	Aantal regels (met Tekstvenster)	Aantal regels (volscherm)	Aantal Kleuren	Vereiste RAM (Bytes) Gedeeld	Vol
0	TEXT	40	—	24	1-1/2		992
1	TEXT	20	20	24	5	674	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

PLOT (PL.) — Zet een enkele punt of teken op een gespecificeerde plaats op het beeldscherm.
Bijv.: PLOT X,Y (X en Y-coördinaten. De X- en Y-waarden dienen positieve getallen te zijn met een waarde geschikt voor de gekozen Graphics modus.)

POSITION (POS.) — Zet de cursor op een bepaalde beeldschermpositie; echter er wordt niets opgetekend. Dit is nuttig o.a. voor het bepalen van de positie van tekst met PRINT.
Bijv.: POSITION X,Y.

LOCATE (LOC.) — Haalt data op van een aangegeven schermlokatie. Haalt tekens in modi 0-2. Haalt COLOR-nummers in de modi 3-11.
Bijv.: LOCATE X,Y,D (Haalt gegevens van schermlokatie X,Y en kent deze aan D toe.)

DRAWTO (DR.) — Trekt een lijn tussen de laatste positie van de cursor en de gespecificeerde X- en Y-coördinaten. De cursor staat dan tenslotte bij de nieuwe coördinaten.
Bijv.: DRAWTO X,Y (Trekt een lijn naar het punt X,Y vanaf de huidige positie van de cursor.)

TABEL VAN SETCOLOR "DEFAULT"* KLEUREN

Setcolor (Kleur-register)	"Default" Kleuren	Helderheid	Weergegeven Kleur
0	2	8	ORANJE
1	12	10	GROEN
2	9	4	DONKER BLAUW
3	4	6	ROSE OF ROOD
4	0	0	ZWART

*"DEFAULT" is een toestand van niet toegekende kleurwaarden, bijv. wanneer geen SETCOLOR-instruktie is gebruikt. In deze toestand neemt BASIC de kleurwaarden aan zoals vermeld onder "Default" kleuren.

NB: De kleuren kunnen, afhankelijk van het gebruikte televisie- of monitor-type, van de aangegeven tabel en kleuromschrijving afwijken.

DE ATARI KLEUREN-(SETCOLOR/INSTRUKTIES)-NUMMERS EN KLEUREN

Kleuren	Setcolor (aexp2) Nummers
GRIJS	0
LICHT ORANJE (GOUD)	1
ORANJE	2
ORANJE-ROOD	3
ROSE	4
VIOLET-ROOD	5
VIOLET-BLAUW	6
BLAUW	7
BLAUW	8
LICHT BLAUW	9
TURQUOISE	10
BLAUW-GROEN	11
GROEN	12
GEEL-GROEN	13
ORANJE-GROEN	14
LICHT ORANJE	15

FUNCTIES

Een functie bewerkt één of meer waarden en geeft de uitkomst. Waarden kunnen strings of getallen zijn. De voorbeelden tonen A (numerieke variabele) of A$ (string variabele) als resultaat van de functie. Een functie kan bijna overal gebruikt worden waar men een waarde zou willen gebruiken, inclusief in een andere functie.

REKENKUNDIGE FUNCTIES

ABS — Geeft de absolute waarde van een getal.
Bijv.: A=ABS (B).

CLOG — Geeft normale logaritme (grondtal 10)
Bijv.: A=CLOG (B).

EXP — Geeft de waarde van e (ongeveer 2.718) tot de aangegeven macht. In enkele gevallen is EXP alleen tot 6 cijfers nauwkeurig. Dit is het tegengestelde van LOG.
Bijv.: A=EXP (B).

INT — Geeft het grootste gehele getal kleiner dan of gelijk aan de ingevoerde waarde.
Bijv.: A=INT (B).

LOG — Geeft de natuurlijke logaritme (grontal e) van een getal. Dit is het tegengestelde van EXP.

RND — Geeft een willekeurig getal tussen 0 en 1. Geeft echter nooit een 1. De waarde maakt geen verschil.
Bijv.: A=RND (0) (A=een getal gelijk aan of groter dan 0 en kleiner dan 1)
A=RND (0)x8 (A=een getal groter dan of gelijk aan 0 en kleiner dan 8).

SGN — Geeft -1 als het getal negatief is, 0 als het getal 0 is, en 1 als het getal positief is.
Bijv.: A=SGN (A).

SQR — Geeft de positieve wortel van een positief getal.
Bijv.: A=SQR(B)

TRIGONOMETRISCHE FUNCTIES

ATN — Geeft de arc tangens van een waarde in radialen of graden.
Bijv.: A=ATN(B)

SIN — Geeft de sinus van een hoek.
Bijv.: A+SIN(B)

COS — Geeft de cosinus van een hoek.
Bijv.: A+COS(B)

DEG — Alle volgende trigonometrische functies worden in graden uitgedrukt.
Bijv.: DEG

RAD — Alle volgende trigonometrische functies worden in radialen uitgedrukt. De computer neemt aan dat waarden in radialen gewenst worden, zolang niet "DEG" gespecificeerd wordt (graden).
Bijv.: RAD

SPECIALE FUNCTIES

ADR — Geeft het decimale geheugenadres van het begin van een string.
Bijv.: A=ADR(BS)
A=ADR ("DEZE STRING")

FRE — Geeft aan hoeveel geheugen (RAM) er nog vrij is. (0) is een verplichte lose variabele.
Bijv.: A=FRE(O)
PRINT FRE(O) (Hiermee wordt op het scherm gezet hoeveel bytes er in het geheugen nog beschikbar zijn)

PEEK — Geeft het getal dat op een specifiek geheugenadres is opgeslagen. Het adres moet een getal tussen 0 en 65535 zijn. Het resultaat zal tussen 0 en 255 liggen.
Bijv.: A:PEEK(B)

USR — Roept op machinetaal subroutines van BASIC. USR(ADDR, P1, P2,..., PN), bij voorbeeld, voegt de argumenten (P1 tot PN) in tegengestelde volgorde bij de stapel. Dientengevolge wordt het laatste argument "PN" aan de stapel toegenvoegd vóór het eerste argument "P1"

Het aantal argumenten — in de vorm van een enkele byte — wordt dan aan de stapel toegevoegd. Als in de functie geen argumenten zijn gespecifierd, dan wordt nul (zero) aan de stapel toegevoegd.

De machinetaal subroutine op address (ADDR) wordt dan opgeroepen. Als de machinetaal routine een waarde in BASIC moet geven, dan moeten de Lage en Hoge bytes in de geheugenlocaties, respectievelijk $D4 en $D5, worden opgeboren.

De routine moet het argumentaantal en de argumenten verwijderen alvorens naar BASIC terug te keren, anders valt het systeem uit elkaar.

Voorbeeld: A=USR(B,C,D)
(De routine bij B zal worden opgeroepen en de parameters in C en D zullen worden doorgegeven aan de subroutine via de stapel).

STRING FUNCTIES

ASC — Geeft de ATASCII-code van het eerste teken van een string terug.
Bijv.: A=ASC ("A") (A wordt 65)
A=ASC (B$) (String variabelen kunnen worden gebruikt. Geeft de ATASCII-waarde van het eerste teken van B$ (B-string)).

CHR$ — Geeft het teken vertegenwoordigd door het ATASCII-nummer. Reciproke functie van ASC.
Bijv.: A$=CHR$ (65) (A$ wordt "A").

LEN — Geeft een getal dat de lengte (aantal tekens) van een string aangeeft.
Bijv.: A=LEN (A$).

STR$ — Geeft een string die een gespecificeerde waarde vertegenwoordigt (Vertaalt een getal in een string).
Bijv.: A$=STR$ (65) (A$ wordt gelijk aan "65", wat door de computer niet meer als een getal herkend wordt, maar als een string).

VAL — Geeft een getal dat in een opgegeven string vertegenwoordigd is. (Vertaalt een string in een getal).
Bijv.: A=VAL ("100") (A wordt gelijk aan het getal honderd).
A=VAL (A$) (A wordt de waarde, vertegenwoordigd in A$).

STRING BEWERKINGEN

ATARI BASIC gebruikt geen string reeks formaat voor string bewerkingen. Een mid-string instruktie staat bewerkingen toe zoals string-verbindingen, delingen en overige handelingen.

Voorbeelden:
String-deling
50 A$ = "JANPIETKLAAS"
60 B$ = A$(9,16) (B$ is "KLAAS")

String koppeling
50 A$ = "HALLO"
60 B$ = "FRED"
70 A$ (LEN (A$ + 1) = B$ (A$ IS "HALLO FRED")

Zoekmogelijkheid in een string
50 FOR Z = 1 TO LEN (A$)
60 IF A$ (Z,Z) = "E" THEN PRINT "AN EZ"
70 NEXT Z

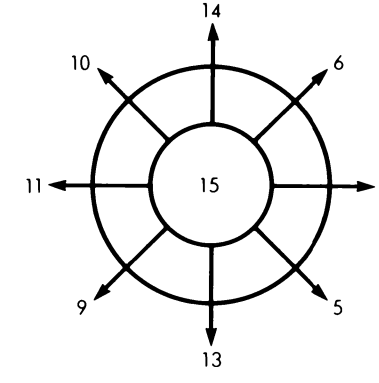
FUNCTIES VOOR STUURKNUPPELS EN DRAAIKNOPPEN

PADDLE — Geeft de waarde van een opgegeven draaiknop-commandopost (Paddle) terug. De paddles zijn van voor naar achter met 0-3 genummerd. Het resultaat ligt tussen 1 en 228 en neemt toe wanneer de knop naar links wordt gedraaid (tegen de klokrichting in).
Bijv.: A=PADDLE(O).

PTRIG — Geeft een 0 als de rode vuurknop van de paddle is ingedrukt en een 1 wanneer deze niet is ingedrukt. De paddles zijn van voor naar achter genummerd van 0 - 3.
Bijv.: A=PTRIG(O)

STICK — Geeft een getal, afhankelijk van de stand van de stuurknuppel (joystick). De joysticks zijn genummerd van 0-1, van voor naar achter. Zie de tekening voor details.
Bijv.: A=STICK(O).

STRIG — Geeft een 0 als een specifieke joystick-vuurknop ingedrukt word en een 1, wanneer deze niet is ingedrukt. De joysticks zijn genummerd 0 en 1.
Bijv.: A=STRIG(O)
Joystick 1 = STICK(O)
Joystick 2 = STICK(1)
De volgende tekening toont de getallen, wanneer de stuurknuppel wordt bewogen in een gegeven richting.



FUNCTIETOETSEN

ESC — Veroorzaakt dat de volgende ingedrukte toets als een internationaal of een grafisch teken voorgesteld wordt en uitgevoerd wordt wanneer een "print" instruktie in een programma wordt tegengekomen.

BREAK — Veroorzaakt dat een BASIC-programma gestopt wordt.

RESET — Stopt een lopend programma, zet het beeldscherm in de Graphics modus 0, en beëindigt uw documentatie zonder deze ar te sluiten. Het programma wordt niet uitgewist.

SET-CLR-TAB — Beweegt de cursor naar de volgende, van te voren vastgelegde, tabulatorstop.

(SHIFT)SET-CLR-TAB — Wist een tabulator stop.

(CTRL SET CLR-TAB) — stelt een tabulator stop in.

(CTRL) 1 — Stopt en start het om hoog rollen van het beeld op het scherm.

(CTRL) 2 — Zoemer

(CTRL) 3 — Duidt het einde van een bestand aan.

SCHERMBEWERKINGS FUNCTIES (EDITING)

SHIFT INSERT — Voegt een regel in op de plaats waar de cursor staat.

CTRL INSERT — Voegt een spatie in op de plaats waar de cursor staat.

DELETE BACK S — Verwijdert het teken links van de cursor, en zet de cursor op deze plaats.

SHIFT DELETE BACK S — Verwijdert de regel waarop de cursor staat.

CTRL DELETE BACK S — Verwijdert het teken waarop de cursor staat en schuift het resterende gedeelte van de regel terug om deze vrijgekomen plaats te vullen.

SHIFT CLEAR of CTRL CLEAR — Wist het scherm.

CTRL-↑ — Beweegt de cursor omhoog.

CTRL-↓ — Beweegt de cursor omlaag.

CTRL-← — Beweegt cursor naar links.

CTRL-→ — Beweegt cursor naar rechts.

FOUTMELDINGEN

CODE	MELDING
2	Onvoldoende geheugen-capaciteit
3	Waarde incorrect
4	Te veel variabelen
5	Fout in de string lengte
6	Beschikbare gegevens niet binnen gevraagd bereik
7	Getal groter dan 32767
8	Fout INPUT-instruktie
9	DIM-fout in gegevensreeks of string
10	Stack vol
11	Drijvende komma getal te groot of te klein
12	Regel niet gevonden
13	Geen passende FOR-instruktie
14	Regel te lang
15	GOSUB- of FOR-regel verwijderd
16	RETURN-fout
17	Syntax-fout (opdracht fout gespeld)
18	Ongeldig teken voor string toepassing
19	LOAD-programma te lang
20	Apparaatnummer groter dan 7
21	LOAD-bestand fout

NB: De volgende fouten zijn INPUT/OUTPUT-fouten, die zich kunnen voordoen bij het gebruik van diskette-stations, printers of ander aangesloten apparatuur. Verdere informatie wordt met de apparatuur verstrekt.

128	BREAK-stop
129	IOCB reeds geopend
130	Nietbestaand apparaat
131	IOCB, alleen schrijven
132	Ongeldige bewerkingsinstruktie
133	Apparaat of bestand niet geopend
134	Verkeerd IOCB-nummer
135	IOCB-"Alleen lezen toegestaan"
136	EOF (Einde van bestand)
137	Ingekorte (verminkte) vermelding
138	Apparaat-timeout foutmelding (reageert niet binnen tijdslimiet)
139	Apparaat NAK foutmelding (Apparaat meldt zich niet beschikbaar)
140	Serial bus foutmelding
141	Cursor buiten toegestaan gebied
142	"Serial Bus Data Frame Overrun" foutmelding
143	"Serial Bus Data Frame Checksum" foutmelding
144	Fout veroorzaakt door extern apparaat
145	Illegale beeldscherm modus—foutmelding
146	Functie niet uitgevoerd
147	Onvoldoende geheugen (RAM)
160	Diskette-station nummer foutmelding
161	Te veel OPEN bestanden
162	Diskette vol
163	Onherstelbaar systeem-gegevens, I/O fout
164	Verkeerd bestandsnummer
165	Bestandsnaam foutmelding
166	POINT lengte van gegevens foutmelding
167	Bestand gesloten
168	Apparaat instruktie ongeldig
169	Disk inhoudsopgave (directory) vol
170	Bestand niet gevonden
171	POINT ongeldig
172	Ongeldig APPEND
173	Fout formaat

Every effort has been made to ensure the accuracy of the product documentation in the manual. However, because we are constantly improving and updating our computer software and hardware, Atari, Inc. is unable to guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors or omissions.

No reproduction of this document or any portion of its contents is allowed without the specific written permission of Atari, Inc. Sunnyvale, CA 94086.

Toutes les measures possibles not été prises afin d'assurer l'exactitude de la documentation du produit dans le manuel. Toutefois, du fait de notre constante amélioration et mise à jour du logiciel et du matériel ("software" et "hardware") de notre ordinateur, Atari, Inc., ne peut garantir l'exactitude de tout matériel imprimé après la date de publication et décline toute responsabilité quant aux changements, erreurs ou omissions.

Ce document ne peut être reprouduit en partie ou en totalité sans l'autorisation écrite expresse de Atari, Inc., Sunnyvale, CA 94086 USA.

Es wurden alle Massnahmen unternommen, um die Richtigkeit der Produktions-dokumentation im Leitfaden zu versichern. Da wir jedoch laufend unsere Software und Hardware verbessern und auf den neuesten Stand bringen, kann Atari, Inc. die Richtigkeit des Druckmaterials nach dem Veröffentlichungsdatum nicht mehr gewährleisten. Atari weist weiterhin darauf hin, dass die Firma für Änderungen, Fehler oder Auslassungen nicht haftet.

Kein Nachdruck dieses Textes bzw. eines Ausschnittes desselben wird ohne die vorherige schrifltiche Genehmigung von Atari, Inc., Sunnyvale, CA 94086 USA, gestattet.

Al het mogelijke is gedaan ten einde in deze handleiding een accurate productomschrijving te verzekeren. Omdat wij echter steeds bezijg zijn onze computer programmatuur en apparatuur te verbeteren en bij de tijd te houden, kan Atari, Inc. niet garanderen dat de literatuur na de publikatiedatum nog correct is en is daarom niet aansprakelijk voor veranderingen, errata of weglatingen.

Reproductie van dit document of enig deel daarvan is niet toegestaan zonder de uitdrukkelijke schriftelijke toestemming van Atari, Inc., Sunnyvale, CA94086 USA.

Se ha hecho todo lo posible para asegurar la exactitud de la documentación del producto en el manual. No obstante, y debido a que continuamente estamos mejorando y modernizando nuestros conjuntos de programas y ordenadores, ATARI, Inc. no puede garantizar la exactitud del material impreso después de la fecha de publicación, y no se hace responsable por cambios, errores u omisiones.

Se prohíbe la reproducción de este documento o de cualquier parte de su contenido sin el consentimiento escrito de ATARI, Inc. Sunnyvale, California 94086.

Ogni cura è stata presa per assicurare l'accuratezza della documentazione in questo manuale. Dato comunque il constante miglioramento e aggiornamento del nostro software e hardware, Atari, Inc. non garantisce l'accuratezza del materiale illustrativo dopo la data di pubblicazione e declina ogni responsibilità dovuta a cambiamenti, errori o omissioni.

Nessuna riproduzione di questo documento, per intero o in parte, è permessa senza l'autorizzazione scritta di Atari, Inc. Sunnyvale, CA 94086, USA.

ATARI and Design, Reg. U.S. Pat. & Tm. Off.
©1983 Atari, Inc.
All Rights Reserved
Printed in Taiwan
C061948 REV. C

ATARI®

ATARI Sales and Distribution Company
P.O. Box 427, Sunnyvale. CA 94086
C.T. 6. 1984