ATARI MICROSOFT BASIC II*

Quick Reference Guide

CONTENTS

Commands	3
Statements	5
Numeric Functions	10
String Functions	11
Special-Purpose Functions	12
Fun Features	13
Error Codes	15

*©1982 MICROSOFT
All rights reserved

PROGRAM COMMANDS

Command	Examples	Brief Summary
AUTO	AUTO AUTO 10,10	Numbers program lines automatically. If only the command is used, the first line is 100 and the increment between line numbers is 10. In the second example, the first line is 10, the second line 20, and so on.
CLOAD	CLOAD	Loads a tokenized BASIC program into memory from cassette tape.
CONT	CONT	Continues the program execution after BREAK or STOP.
CSAVE	CSAVE	Stores a program in memory to a cassette tape in tokenized format.
DEL	DEL 10 DEL 10-100 DEL 10- DEL -10	Deletes one or a range of lines from a program. Use hyphens (-) to determine range.
DOS	DOS	Calls the Disk Operating System menu. To return to Microsoft BASIC II, use the "B" option.
KILL	KILL "D:PROG1.AMB"	Deletes a program from the named device. In the example, the file "PROG1.AMB" is deleted from the diskette in the disk drive.
LIST	LIST LIST 10 LIST 10-100 LIST 10- LIST -10 LIST "P:" LIST "C:"	Lists one, a range, or all program lines to the TV screen. Hyphens (-) are used to set the range. Also lists program lines on the printer, cassette, or other specified device.
LOAD	LOAD "D:EXAMPLE1.AMB" LOAD "C:"	Loads a program into the computer's memory. Program can be on diskette or cassette.
LOCK	LOCK "D:PROGRAM2.AMB"	Protects the program on diskette from accidental erasure.
MERGE	MERGE "D:EXAMPLE2.AMB" MERGE "C:"	Appends a program on diskette or cassette to one in memory. All duplicate line numbers coming from the device replace those in memory.

PROGRAM COMMANDS

Command	Examples	Brief Summary
NAME...TO	NAME "D:OLDFILE" TO "NEWFILE"	Renames a program on a device. In the example, the file named "OLDFILE" is renamed "NEWFILE."
NEW	NEW	Erases a program from memory.
RENUM	RENUM RENUM 100,50,10	Renumbers program lines. If no parameters are used, the first line of the program is changed to 10, and the rest of the lines are incremented by 10. In the second example, program line 50 is changed to 100 and the rest of the lines are incremented by 10.
RUN	RUN RUN 100	Begins execution of a program. In the second example, program execution begins at line 100.
SAVE	SAVE "D:TESTFILE.AMB"	Saves a program to a device in tokenized format.
SAVE...LOCK	SAVE "D:TESTFILE.AMB" LOCK	Saves a program to diskette in tokenized format and locks it to prevent tampering. You cannot list or modify a locked file.
TROFF	TROFF	Turns off the trace mechanism.
TRON	TRON	Turns on the trace mechanism.
TROFF	TROFF	Turns off the trace function.
TRON	TRON	Turns on the trace function.
UNLOCK	UNLOCK "D:TESTFILE.AMB"	Unlocks a file on diskette so that you can write to, delete, or rename it.
VERIFY	VERIFY "D:FILEPROG.AMB" VERIFY "C:"	Compares two programs–the one in memory with one on diskette or cassette. If they do not match exactly, a TYPE MISMATCH ERROR occurs.

PROGRAM STATEMENTS

Statement	Examples	Brief Summary
AFTER	AFTER (3600) 125	Starts a time count using jiffies (1/60 of a second). In the example, after 3600 jiffies (1 minute), the program continues at line 125.
CLEAR	CLEAR	Zeros all variables, nulls all strings, and clears all arrays.
CLEAR STACK	CLEAR STACK	Clears all time counts. May be used to abort the AFTER statement.
CLOSE	CLOSE #1	Closes a previously opened file. The # sign is mandatory with the number to identify the input/output control block.
COMMON	COMMON ALL COMMON L, L1(2), L$	Keeps program variables intact from one program to another. You may retain the contents of one, many, or all variables from program to program.
DEF	DEF AVG(X,Y)=(X+Y)/2	Allows you to define your own functions. Both number and string functions may be defined. User-defined string functions are only available when the extension diskette is used.
DIM	DIM A$(35) DIM NUM(10,5,2)	Dimensions arrays. DIM tells the computer the number of elements expected in a string or numerical array. An array may be multi-dimensional.
END	END	Terminates a program. It is the last statement used in a program. Closes all files and clears all time counts.
ERROR	ERROR 2	Forces an error in a program (as a debugging measure) to test how a program behaves when an error occurs. Forces both SYSTEM and BASIC errors.
FOR...TO...STEP /NEXT	FOR X=0 TO 10	Sets up a counter for repeated execution of one or a group of statements. Executes all statements before the NEXT command, until the counter reaches the TO number. If STEP is used, the counter increments by the STEP amount.
GET	GET #1,X GET #1, AT(8,2)	GET and PUT are opposites. GET reads a single byte value and stores it in a variable.
GOSUB /RETURN	GOSUB 100	Causes a program to jump to another line and later return to the next statement. Used for calling a subroutine.
GOTO	GOTO 50	Causes a program to jump to another line to continue execution.

PROGRAM STATEMENTS

Statement	Examples	Brief Summary
IF...THEN	IF X=1 THEN Y=X	Tests strings and numbers for true and false conditions. If a condition is true, the program carries out the command following the THEN statement. If the condition is false, the program continues execution at the next line.
IF...THEN... ELSE	IF X=0 THEN Y<>X ELSE Y=X	Is the same as IF...THEN except that a false condition means program execution passes to the command following the ELSE statement.
INPUT	INPUT I INPUT I$ INPUT "Type a number";I INPUT "Your name?";I$	Halts program execution to accept information from another device (default is keyboard). The INPUT statement accepts assignments to string and number variables. A "?" is used as a prompt unless the string option is used.
INPUT...AT	INPUT AT (2,4) I$	Is the same as INPUT except that the program accepts input at a specific location (column, row) on the TV screen.
LET	LET Z=2	Is optional for variable assignments. Z=2 is also acceptable.
LINE INPUT	LINE INPUT "Name: "; I$	Is the same as INPUT except that a full line (including spaces, commas, colons, and other delimiters) may be input from the keyboard or specified device. A "?" is used as a prompt unless the string option is used.
LINE INPUT...AT	LINE INPUT AT (4,4) I LINE INPUT AT (6,8) "Name: ";I$	Is the same as LINE INPUT except program accepts input at a specific location (column, row) on the TV screen.
MOVE	MOVE 55,222,5	Moves memory from one location to another. In the example, five bytes of memory starting at address 55 (decimal) are moved to the address beginning at 222. Hexadecimal numbers can also be used.
NEXT	NEXT NEXT I	Ends a FOR...TO...STEP statement block. The variable name is optional. (See FOR... TO...STEP/NEXT.)
NOTE	NOTE #4, I,J	Locates the next byte to be read from a diskette file. In the example, NOTE stores the position of the current sector number in I and the current byte in J.

PROGRAM STATEMENTS

Statement	Examples	Brief Summary
ON ERROR	ON ERROR 550	Forces execution of a program to a specified line when encountering an error. In the example, the program will continue at line 550 if an error occurs. RESUME is required to return execution to the original routine.
ON...GOSUB /RETURN	ON G GOSUB 100,200,300	Determines which subroutine to execute next. In the example, the variable G should be 1, 2, or 3, causing the program to jump to line 100, 200, or 300. (See RETURN.)
ON...GOTO	ON G GOTO 100,200,300	Determines which of a group of lines will be executed next. In the example, G should be 1, 2, or 3, causing program execution to jump to line 100, 200, or 300.
OPEN	OPEN #3,"P:" OUTPUT OPEN #4,"D:PROGSAV.AMB" INPUT	Opens a file for reading or writing. The statement identifies the input/output control block used by a specified device, such as a printer or disk drive, and declares the type of operation to be carried out (UPDATE, APPEND, INPUT, or OUTPUT).
OPTION BASE	OPTION BASE 1	Declares the base number of all arrays and variables. Allows the user to set the base number used in loops and arrays. The default is zero (0). In the example, all program arrays will automatically begin at 1.
OPTION CHR	OPTION CHR1 OPTION CHR2 OPTION CHR0	Reserves bytes of memory for RAM character data. OPTION CHR1 sets aside 1024 bytes, OPTION CHR2 sets aside 512 bytes, and OPTION CHR0 releases all OPTION CHR reservations. (See VARPTR.)
OPTION PLM	OPTION PLM1 OPTION PLM2 OPTION PLM0	Reserves bytes of memory for player-missile graphics. OPTION PLM1 reserves 1024 bytes, OPTION PLM2 reserves 512 bytes, and OPTION PLM0 releases all OPTION PLM reservations. (See VARPTR.)
OPTION RESERVE	OPTION RESERVE 24	Reserves bytes of memory for machine language routines. (See VARPTR.)
PRINT	PRINT"Hello" PRINT 25*4 PRINT "Reply=";R$?Y	Prints number and string constants and variables on the TV screen. By itself, PRINT causes a blank line to be printed on the TV screen. The question mark symbol (?) can also be used in place of the full word PRINT.
PRINT...AT	PRINT AT(4,4)RX$	Prints number or string constants and variables at a specific location (column, row) on the TV screen, or a specific sector and byte of a diskette file.

PROGRAM STATEMENTS

Statement	Examples	Brief Summary
PRINT...SPC	PRINT SPC(5)"Hi!";SPC(5)"Bye"	Prints the number of spaces specified in the parentheses, counting from the current cursor position. Differs from TAB, which always counts spaces from the leftmost column.
PRINT...TAB	PRINT TAB(5)"Hello"	Prints the number of spaces specified in the parentheses, starting at the leftmost column of the text field.
PRINT USING	PRINT USING "#";1 PRINT USING "##";NUMBER PRINT USING "###.##";MONEY PRINT USING "###,###.##";AMT# PRING USING "** ###.##";CASH PRINT USING "$###.##";DOLLAR PRINT USING "$$###.##";CHECK PRINT USING "**$###.##";FLOAT PRINT USING "##^^^^";EXPONENT PRINT USING "+###";PLUS PRINT USING "###-";MINUS PRINT USING "!";INITIAL$ PRINT USING "% %";PART$	Lets you format your text 12 ways, including: —Aligns numbers in columns signified by pound sign —Places a decimal point in the result —Offsets every three digits (thousands) with a comma —Pads empty digit spaces with asterisks —Prints a dollar sign ($) before left digit —Prints a floating dollar sign ($) in result —Combines floating "$" with filler "*" in result —Prints result in exponential (E or D) format —Prints a plus sign (+) before or after result —Prints a minus sign (—) before or after result —Pulls out the first character in a string —Pulls out part of a string
PUT	PUT #6, ASC("A");	PUT and GET are opposites. PUT writes a single byte value (0-255) to a specified file or device.
RANDOMIZE	RANDOMIZE RANDOMIZE 20	Seeds the RND function to assure that a different sequence of random numbers occurs each time a program is run. The second example assures that a random sequence is generated repeatedly.
READ/DATA	READ A,B,C DATA 1,2,3	Assigns numbers or strings in the DATA statement to variable names in the READ statement.
REM or ! or '	REM Ignore this comment ! Ignore this remark ' Ignore this remark, also X=1:REM Colon is necessary X=1 ' No colon is necessary	Allows explanations in your program. REM (short for "remark") statements are ignored during program execution. You can use an exclamation mark (!) or an apostrophe (') in place of the word REM.

PROGRAM STATEMENTS

Statement	Examples	Brief Summary
RESTORE	RESTORE RESTORE 110	Allows the reuse of DATA statements. If no parameter is used, READ re-reads data from the first DATA statement. In the second example, RESTORE causes the READ statement to begin reading data at line 110.
RESUME	RESUME RESUME 55 RESUME NEXT	Helps program recover from an error or a time count. RESUME sends program execution back to the line in which an error or time interrupt occurred. If a line number is used, the program resumes at that line. If a NEXT statement is used, program execution resumes with the statement following the error or interrupt (may be on the same line). RESUME completes the ON ERROR and AFTER statements.
RETURN	RETURN	Completes the GOSUB and ON...GOSUB statements and returns a program from a subroutine. (See GOSUB.)
STACK	PRINT STACK IF STACK=0 THEN PRINT "Stack full"	Gives the number of entries available on the time stack (used to hold jiffies for AFTER and SOUND).
STOP	STOP	Halts execution of the program. Use CONT to continue program execution (starting with the next line).
WAIT...AND	WAIT &D40B,AND &FF,110	Halts program to wait for certain conditions to occur. Advanced animation techniques use WAIT to handle VBLANK. In the example, the program looks at the contents of address &D40B, ANDs it with &FF, and waits until it equals 110.

PROGRAM FUNCTIONS

Numeric Functions	Examples	Brief Summary
ABS	Y=ABS(−7)	Computes the absolute value of a number.
ATN	X=ATN(5.3)	Computes the arctangent of a number.
COS	PRINT COS(.95)	Computes the trigonometric cosine of a number.
EXP	EULER=EXP(3)	Computes the Euler's number (e) raised to the power of the number in parentheses.
INT	PRINT INT(5.3)	Returns the integer of a number, always rounding down to the next lower number.
LOG	L=LOG(.5)	Computes the natural logarithm of a positive, nonzero number.
RND	R=RND PRINT RND(0) NUMBER=RND(100)	Generates random single-precision numbers. RND by itself or with a zero in parentheses produces a random value between 0 and 1. Used with a nonzero number in parentheses, it produces an integer between 1 and the number. In the example, RND (100) returns a random number between 1 and 100.
SGN	PRINT SGN(R*B)	Returns the sign of a number. If the number is positive, the value returned is +1. If the number is zero, the value returned is 0. If the number is negative, the value returned is −1.
SIN	PRINT SIN(1)	Computes the trigonometric sine of a number.
SQR	ROOT=SQR(25)	Returns the square root of a positive number.
TAN	PRINT TAN(.22)	Computes the trigonometric tangent of a number.
+ (Concatenation)	PRINT INITIAL$ + NAME$	Joins two strings together.
ASC	PRINT ASC("Sam")	Returns the ATASCII code in decimal for the first character contained in parentheses. In the example, the first character is "S," which is 83 (decimal) in ATASCII code.
CHR$	A$=CHR$(65)	Converts the ATASCII code in parentheses to a one-character string. CHR$ is the opposite of the ASC function.
INKEY$	R$=INKEY$	Returns the last key pressed.

PROGRAM FUNCTIONS

String Functions	Examples	Brief Summary
INSTR	HOLD=INSTR(5,A$,"THE")	Searches for a small string inside a large string. Returns the character position within the larger string where the smaller string begins. If not found, it returns a zero (0). In the example, the search begins at the fifth character in A$, looking for "THE." If no starting number is given, the search begins at the first character in the larger string.
LEFT$	PRINT LEFT$("LEFTY",4)	Returns characters from the left side of a string. In the example,"LEFT" will be printed.
LEN	PRINT LEN(C$)	Returns the length (number of characters) of a string.
MID$	PRINT MID$("THEMIDPART",4,3)	Returns characters from the middle part of a string. In the example, the selection of three characters in A$, starting with the fourth character, result in printing "MID."
RIGHT$	A$=RIGHT$("THERIGHT",5)	Returns characters from the right side of a string. In the example, the A$ is assigned "RIGHT."
SCRN$	C$=SCRN$(5,5) PRINT ASC(C$)	Returns the value of the character in text modes; in graphics modes, it returns the color register number in ATASCII code (except in graphics modes 4 and 6).
STR$	X=STR$(99.99)	Converts a number into a string. It is the opposite of VAL.
STRING$	PRINT STRING$(36,"*") PRINT STRING$(36,42)	Returns a string of characters. In the first example, 36 repetitions of a string (the asterisk) are printed. In the second example, 36 repetitions of the CHR$ function are printed (42 is the value of the asterisk in ATASCII code). In other words, both result in a string of 36 asterisks.
TIME$	PRINT TIME$	Returns the contents of the TIME$ string, which consists of hours, minutes, and seconds (12:01:00).
VAL	PRINT VAL(R$)	Converts a string to a number. It is the opposite of STR$.
EOF	EOF (4)	Returns a value of true or false to indicate the end-of-file condition of the last read from an input/output control block.

PROGRAM FUNCTIONS

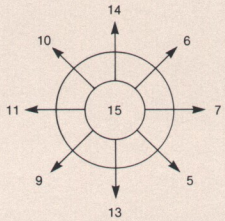
Special Purpose	Examples	Brief Summary
ERL	PRINT ERL	Returns the line number of the last encountered error.
ERR	PRINT ERR	Returns the code number of the last encountered error.
FRE (0)	PRINT FRE(0)	Returns the number fo free RAM bytes available for use.
PEEK	PRINT PEEK(751) ADRS=PEEK(PLACE)	Returns the contents of the memory address enclosed in parentheses. The address and byte can be a number or variable, in decimal and hexadecimal. You may look at ROM as well as RAM addresses.
POKE	POKE 1034,255 POKE ADRS,J POKE PLACE,X*Y/2	Inserts a byte into an address location in RAM (but not in ROM). The address and byte can be a number, arithmetic expression, or a variable—in decimal or hexadecimal.
STATUS	PRINT STATUS (4)	Holds the value of the fourth byte of the input/output control block; the value tells the error condition (1=error; 0=no error).
TIME	PRINT TIME	Returns the real time clock (RTCLOK) location's contents. Unlike TIME$ (which returns the elapsed time in hours, minutes, and seconds), TIME gives its value in jiffies (1/60 of a second).
USR	X=USR(898,0)	Passes control of the program to a machine language subroutine. In the example, the USR function passes control to a subroutine at 898 decimal. An optional number may be passed along with the address, for use in the subroutine.
VARPTR	ADRS=VARPTR(A$) PRINT VARPTR(A$)+1 PLAYR1=VARPTR(PLM1) PRINT VARPTR(CHR1) PRINT VARPTR(RESERVE)	Returns the memory address of a variable's symbol table entry. In the first example, VARPTR returns the number of bytes in the string. In the second example, the starting address of the string is printed. In the other examples, VARPTR returns the address (MSB,LSB) of the first byte allocated for player-missile graphics, a character set, and the reserved memory set aside for assembly language programs.

FUN FEATURES

Graphics	Examples	Brief Summary
CLS	CLS CLS 25	Clears the screen text areas and sets the background color register to the indicated color value. In full screen modes, the optional number after CLS sets the border color and luminance. In split screen it determines the background color and luminance.
COLOR	COLOR 4	Specifies the color register to be used for color graphics.
FILL	FILL 5,5 TO 5,20	Fills an area of the TV screen with the color of the specified color register number.
GRAPHICS	GRAPHICS 0 GRAPHICS 2+16 GRAPHICS 7+32	Selects one of 12 graphics modes (CTIA provides 8 graphics modes). Adding +16 to any graphics mode provides a full screen display. Adding +32 prevents the graphics command from clearing the screen.
PLOT PLOT...TO	PLOT X,Y PLOT 5,5 TO 10,5	Draws a point, a line, or several continuous lines on the TV screen.
SETCOLOR	SETCOLOR 5,4,10	Associates a color and luminance with a color register. The first parameter names the register used (0-3 is for player-missiles, 4-7 is for playfield colors, and 8 is always the background register). The second parameter is the color hue number (0-15), while the third parameter is the luminance (an even number between 0 and 14 – the higher the number, the brighter the luminance).
Paddle Controllers (4 sets/2 per set)	PADDLE(0) = 624 PADDLE(1) = 625 PADDLE(2) = 626 PADDLE(3) = 627 PADDLE(4) = 628 PADDLE(5) = 629 PADDLE(6) = 630 PADDLE(7) = 631	Paddle controller statuses are stored in locations 624 to 632. There are four paddle controller ports, each port handling two paddle controllers. The leftmost paddle controller's status is stored in location 624, and so on. The status ranges from 1 to 228 as the paddle is turned counterclockwise.

FUN FEATURES

Game Controllers	Number	PEEK Location	Brief Summary
Paddle Triggers (4 sets/2 per set)	PTRIG(0) PTRIG(1) PTRIG(2) PTRIG(3) PTRIG(4) PTRIG(5) PTRIG(6) PTRIG(7)	=636 =637 =638 =639 =640 =641 =642 =643	Paddle trigger statuses are in locations 636 to 643. The trigger of the leftmost paddle is stored in location 636, and so on. If the trigger has been pressed, a zero (0) is found at its address by PEEK(PTRIG(n)). If the trigger has not been pressed, it contains a one (1).
Joystick Controllers	STICK(0) STICK(1) STICK(2) STICK(3)	=632 =633 =634 =635	The joystick controller statuses are stored in locations 632-635. The leftmost joystick status is stored in location 632, and so on. The status contains a value as shown in the figure below.
Joystick Triggers	STRIG(0) STRIG(1) STRIG(2) STRIG(3)	=644 =645 =646 =647	The joystick trigger statuses are stored in locations 644 to 647. The trigger status of the leftmost joystick is stored in location 644, and so on. The status byte contains a zero (0) if the trigger has been pressed, and a one (1) if not.
Console Keys	OPTION SELECT START No Key	→ PEEK(53279)=3 → PEEK(53279)=5 → PEEK(53279)=6 → PEEK(53279)=7	The status of the OPTION,SELECT, and START keys is stored in location 53279. The status contains a 7 until one of the three keys is pressed. It then has a value of 3, 5, or 6, depending on which key is pressed.



ERROR CODES

For a full explanation of the following errors, see Appendix O of the ATARI Microsoft BASIC II Reference Manual:

Error Code #	Explanation		
1	NEXT without FOR	135	IOCB read-only error
2	Syntax error	136	EOF
3	RETURN without GOSUB	137	Truncated record
4	Out of data	138	Device timeout
5	Function call error	139	Device NAK
6	Overflow	140	Serial bus
7	Out of memory	141	Cursor out of range
8	Undefined line	142	Serial bus data frame overrun error
9	Subscript out of range		
10	Redimension error	143	Serial bus data frame checksum error
11	Division by zero		
12	Illegal direct	144	Device-done error
13	Type mismatch	145	Read after write-compare error
14	File I/O error		
15	Quantity too big	146	Function not implemented
16	Formula too complex	147	Insufficient RAM
17	Can't continue	160	Drive number error
18	Undefined user function	161	Too many OPEN files
19	No RESUME	162	Disk full
20	RESUME without error	163	Unrecoverable system data I/O error
21	FOR without NEXT		
22	LOCK error	164	File number mismatch
23	Time error	165	File name error
128	BREAK abort	166	POINT data length error
129	IOCB error	167	File locked
130	Nonexisent device	168	Command invalid
131	IOCB write-only error	169	Directory full
132	Invalid command	170	File not found
133	Device or file not open	171	POINT invalid
134	Bad IOCB number		

©1982 Atari, Inc.
All rights reserved

A Warner Communications Company

PRINTED IN U.S.A.
C061253 REV. A