Guide to

The BASIC 4.5

For the Commodore C64

Tümmers, Robert
10.6.2021

C64 BASIC 4.5 MANUAL

a BASIC extension for the C64

by Janne Peräaho & Anders Persson – the BASIC 3.5 part
by Robert Tümmers (DG5KR) – the BASIC 4.5 part

Table of contents

Introduction 2
COMMANDS and STATEMENTS 3
Functions 11
Operators 13
Reserved Variables 13
BASIC Error Messages 14
DISK Error Messages 16
PETascii codes 19
Musical note table 20
Special particularities of BASIC 4.5 21
TEDMON 22
ESCAPE CODES 24
BASIC 4.5 Tokenlist 25
Room for additions 26

Introduction

Basic is a high level language which is based on the following six concepts: commands, statements, functions, variables, operators, and expressions.

BASIC 4.5 is a merged product from M&T 64er, 1990/06 –„Ein basic für alle“ by Michael Schimek (BASIC 3.5 on C64). The BASIC 4.5 -Extend the BASIC 3.5 for C64 by Robert Tümmers (DG5KR) in 2020.
Commands and statements are instructions to the computer to perform a certain task (for FORMAT an instruction to load a basic program into memory). The difference between them is that Basic commands are intended to be used in direct mode, while statements should be used in programs. However, in most cases commands can be used as statements in a program if you prefix them with a line number. You can also use several statements as commands by using them in direct mode (i.e. without line numbers).
A function performs a simple task, based on a given argument, and it always replies with a value - a result.
Operators are used for calculations, for determining equalities/inequalities, and for logical operations. For FORMAT + is an operator used for addition.
Expressions are clauses composed of constants, variables, and/or operators. For FORMAT A+B*3 is a valid expression.
This manual's purpose is to provide detail information about presented Basic elements. I hope you find it useful.

COMMANDS and STATEMENTS

Command	Description	FORMAT or EXAMPLES	D/P*
‘	1. Attaches a note to the source code. 2. The ‘ is the target for CALL and JUMP command. Replaces GOSUB and GOTO.	‘ Notes and Text ‘TARGET NAME	
AT	position the cursor on screen	AT 0,20 - Set the cursor in row 0 and column 20.	
AUTO	automatic line numbering	AUTO 10 Automatically numbers line in increments of ten.	D
BACKUP	Copies all the files on a disk to another disk	BACKUP DO TO D1, ON U9 Copies all files from drive 0 to drive 1 in disk drive unit 9.	
BEGINBLOCK	Define a Command block. The block executes only if condition is true. (See also ENDBLOCK)	BEGINBLOCK [any condition] INPUT “ENTER STRING: “;A$ BEGINBLOCK A$=”E” :PRINT “EXECUTE BLOCK IF A$<>E” ENDBLOCK	
BEEP	Sounds a short system beep	BEEP	
BOX	Draws a rectangle	BOX [<color_src>],<left>,<top>[,<right>,<bottom>][,<angle>[,<fill_flag>]	
CALL	Acts like a GOSUB. A name (max. 16 characters) is used instead of a line number. The goal is the apostrophe with the name. A RETURN jumps back and jumps to the next line.	CALL “TARGET” or T$=”TARGET”:CALL T$ The target looks like this: ‘TARGET [Next line…]	
CATALOG	List the disk directory page by page in a user friendly format for direct cursor position and operation.	CATALOG [U]9…[U]11	
CHAR	Prints string on a screen	CHAR [<color_src>],<left>,<top>,"<string>"[,<reverse_flag>]	
CIRCLE	Draws a circle, ellipse, arc, triangle or an octagon	CIRCLE [<color_src>][,<x>,<y>],<x_radius>[,[<y_radius>][,[<s_angle>]	
CLOSE	Closes an open logical file	CLOSE <file nr>	
CLR	Erases any variables in memory	CLR	

CLS	Clear the screen and delete all set screen window. SCNCLR cleans only inside a screen window. CLS unset the screen window and clear the hole screen.	CLS	
CMD	Redirects output	CMD <l_file>[,<w_list>]	
COLLECT	Deletes references to improperly closed files	COLLECT D0 ON U8	
COLOR	Assigns a color to the color source	<color_src>,<color>[,<luminance>]	
COPY	Copies a file	COPY [D0,] "FILEN" TO [D1],"FILENEW"[,U9]	
CONT	Re-start the execution of a program that has been stopped	CONT	D
DATA	Declares data items	DATA <item>[[,<item>][,<...>[,<item>]]]	P
DEF FN	Defines a function	DEF FN <fnc_name>(<variable>)=<expression>	P
DELAY	Hold the execution for 1/50 (NTSC:1/60) * N seconds	DELAY [N] N is an 16 bit unsigned integer number	
DELETE	Deletes lines of BASIC text	DELETE [<first_line>][-<last_line>]	
DIM	Presents and reserves memory for an array	DIM [<variable>(<subscripts>)][,<...>]]	P
DIRECTORY	Displays a disk directory	DIRECTORY [D0][,U8][,"<file>"] (See CATALOG on 7)	
DLOAD	Loads a program from disk into a memory	DLOAD "<file>"[,D<drive>][,U<unit>]	
DLRUN	Load and run a basic program from disk.	DLRUN "PROGNAME"	
DO	Defines a program loop	DO [UNTIL <bool_arg> | WHILE <bool_arg>] <statements> [EXIT] LOOP [UNTIL <bool_arg> | WHILE <bool_arg>]	P
DRAW	Draws dots, lines, and shapes	DRAW [<color_src>][<x>,<y>,][[,]TO <x>,<y>][,<...>[,<x>,<y>]]	
DSAVE	Stores a program on disk	DSAVE "@<file>"[,D<drive>][,U<unit>] @ = OVERWRITE PROGRAM ON DISK	
END	Stops program execution	END	
ENDBLOCK	Set the end of a BEGINBLOCK (See also BEGINBLOCK)	ENDBLOCK	

FILES	Read all PRGs from directory and store then into a string array	FILE V$(0),I,[U]8 10 cls 20 clr 30 files v$(0),i,8 40 print "count of files:",i 50 for x=1 to i 60 print v$(x) 70 next x	
FIND	Find BASIC Command or any Text and List line number or BASIC line.	FIND GOTO,1 - show all lines with the GOTO Command. FIND RETURN - show only line number with RETURN. FIND "AUTOR"- Show all Lines with the text "AUTOR"	
FOR	Defines a program loop	FOR <loop_var>=<start_val> TO <end_val> [STEP <increment>]	
FRAME	Draw a text frame	FRAME X1,Y1,X2,Y2,FRAME COLOR,FRAME TYPE [0-2],"TEXT"	
GET	Gets data from the keyboard	GET <variable>	
GET#	Gets data from a file or a device	GET# <file>,<variable>	
GETKEY	Gets data from the keyboard	GETKEY <variable>	
GOSUB	Calls a subroutine	GOSUB <line>	
GOTO	Redirects program execution	GOTO <line>	
GRAPHIC	Changes graphic mode	GRAPHIC <mode>[,<clr_flag>] 0 normal text 1 high-resolution graphics 2 high-resolution graphics, split screen 3 multicolor graphics 4 multicolor graphics, split screen <clr_flag> - screen clear flag (0=off, 1=on)	

GSHAPE	Displays a shape on a graphic screen	GSHAPE <shape>[,[<x>,<y>][,<mode>]] <shape> - string variable containing a shape to be drawn <x> - scaled x coordinate. The default display position is the pixel cursor <y> - scaled y coordinate. The default display position is the pixel cursor <mode> - replacement mode (0-4) 0 place shape as is (default) 1 place field inverted shape 2 OR shape with area 3 AND shape with area 4 XOR shape with area	
HEADER	Formats a disk	HEADER "<diskname>",D<drive>[,I<id>][,ON U<unit>]	
HELP	Displays the erroneous program line	HELP	D
IF	Conditional execution	IF <expression> THEN: <clause> [:ELSE <clause>]	
INFO	Create a One-Pager info side in a editor	INFO This command will create or modify a short mem. The memo is restricted for one side (40+25 chars) and stored in file named "memo.txt" as SQR file. If the file does not exist, it will be create on last use drive. The minimal editor supports the BASIC 4.5 ESC-commands describe on page 24 (ESCAPE CODES). In future, the command will be renamed from INFO to MEMO with the same token number.	
INPUT	Asks input from the user and stores acquired data	INPUT["<prompt>";]<variable>[,<...>,<variable>]	
INPUT#	Reads data from a file or a device	INPUT#<file>,<variable>[,<...>,<variable>]	
KEY	Assigns a string into a function key	KEY [<key>,<string>]	
LET	Lets a var to assign a value	LET var=value (LET is optional)	
LIST	Lets you look at lines of a BASIC program	LIST [<first_line>][-[<last_line>]]	
LOAD	Loads a program from storage device into a memory	LOAD ["<file>"[,<device>][,<rel_flag>]]	

LOCATE	Changes graphic pixel cursor position	LOCATE <x>,<y>	
LTRIM	Deletes all spaces at the beginning of a string	LTRIM X$	
MEMORY	Show used and free memory of the c64.	MEMORY - Shows detail memory of program, vars an arrays MEMORY X - Store free memory to X	
MONITOR	Starts machine language monitor	For detail information see chapter TEDMON	
NEW	Erases BASIC program in memory	NEW	
NEXT	Completes a FOR loop	NEXT [<variable>[,<...>,<variable>]]	
ON	Redirects program execution conditionally	ON <expression> GOSUB <line>[,<...>,<line>] ON <expression> GOTO <line>[,<...>,<line>]	
OPEN	Opens a logical file for I/O operations	OPEN <file>[,<device>[,<address>[,"<command>,<type>,<mode>"]]] INPUTS <file> - logical file number for the file to be opened (1-255) <device> - input/output device number <address> - secondary address for device <command> - command for device <type> - file type (prg/seq/rel/usr) <mode> - I/O mode (read/write) Device numbers: 1 Keyboard 3 Screen 4 Printer 8-11 Disk 15 Command channel	
PAINT	Fills an area with color	PAINT [<color_src>][,[<x>,<y>][,<mode>]]	
PRINT	Writes data to the screen	PRINT <printlist>[;]	

<table>
<tr><td>PRINT USING</td><td>Formats and writes data to the screen, file or device</td><td>PRINT[<file>,]USING <formatlist>;<printlist>
Formatlist:
<table><tr><th>Description</th><th>Char</th><th>Numeric</th><th>String</th></tr><tr><td>Hash Sign</td><td>#</td><td>X</td><td>X</td></tr><tr><td>Plus</td><td>+</td><td>X</td><td>-</td></tr><tr><td>Minus</td><td>-</td><td>X</td><td>-</td></tr><tr><td>Decimal Point</td><td>.</td><td>X</td><td>-</td></tr><tr><td>Comma</td><td>,</td><td>X</td><td>-</td></tr><tr><td>Dollar Sign</td><td>$</td><td>X</td><td>-</td></tr><tr><td>Four Carets</td><td>^^^^</td><td>X</td><td>-</td></tr><tr><td>Equal Sign</td><td>=</td><td>-</td><td>X</td></tr><tr><td>Greather Than Sign</td><td>></td><td>-</td><td>X</td></tr></table></td><td></td></tr>
<tr><td>PRINT#</td><td>Writes data to a file or a device</td><td>PRINT#<file>,<printlist></td><td></td></tr>
<tr><td>PUDEF</td><td>Redefines PRINT USING symbols</td><td>PUDEF "<definition>"
<definition> - definition string for symbols (from left to right): the first character defines a filler character, the second character defines a comma, the third character defines a decimal point, and the fourth character defines a dollar sign.
10 PUDEF "*" - Prints * in the place of blanks.
20 PUDEF " &" - Prints & in the place of commas.
30 PUDEF " ," - Prints decimal points in the place of commas, and commas in the place of decimal points.
40 PUDEF " .,£" -Prints English pound sign in the place of $, decimal points in the place of commas, and commas in place of decimal points.</td><td></td></tr>
<tr><td>POKE</td><td>Writes a value into a RAM memory</td><td>POKE 16Bit memory,8Bit data</td><td></td></tr>
<tr><td>READ</td><td>Get information from DATA statements</td><td>READ <variable>[,<...>,<variable>]</td><td></td></tr>
<tr><td>REM</td><td>Attaches a note to the source code</td><td>REM Notes and Text</td><td></td></tr>
</table>

RENAME	Renames a file	RENAME [D<drive>,]"<old_filename>" TO "<new_filename>"[,U<unit>]	
RENUMBER	Renumbers program lines	RENUMBER [<new_line>[,<increment>[,<start_line>]]]	D
RESTORE	Reset the DATA Pointer (See RESTORE above) or enable | disable the RESTORE KEY. By disable the RESTORE KEY the key is not able to interrupt a running Program or jumping into the build in Monitor	RESTORE [ON | OFF]	
RETURN	Returns from a subroutine	RETURN	
RTRIM	Deletes all spaces at the end of a string	A$="TEXT " RTRIM A$ PRINT A$ Output: "TEXT"	
RUN	Executes a program	RUN [<line>]	
SAVE	Stores program in a storage device	SAVE [<file>[,<device>[,<eot_flag>]]]	
SCALE	Controls bit maps scaling	SCALE 0 | 1 SCALE 1 - turns scaling on. Coordinates may then be scaled from 0 to 1023 in both x and y rather than the normal scale values, which are: multicolor mode: x = 0 to 159, y = 0 to 199 high resolution mode: x = 0 to 319, y = 0 to 199	
SCRATCH	Deletes a file from disk	SCRATCH "<file>"[,D<drive>][,U<unit>]	
SCNCLR	Clears the screen	Clears the current screen, whether graphics, text, or both (split screen).	
SOUND	Produces a sound	SOUND <voice>,<frq_control>,<duration> VOICE No 1-2 sounds square VOICE No 3 sounds noise For more information see chapter Musical note table	
SSHAPE	Saves a rectangular graphic area into a string variable	SSHAPE <shape>,<left>,<top>[,<right>,<bottom>] SSHAPE V$,0,0 - Saves screen area from the upper left corner to where the cursor is positioned under the name V$.	

STOP	Stop a running program or enable|disable the STOP KEY	STOP [ON|OFF] STOP OFF - RUN/STOP KEY DISABLED	
SYS	Executes a machine language program	SYS 16Bit Address Parameters can be passedanyway using the following memory locations: 2034 ($07F4)= Accumulator 2035 ($07F5)= X register 2036 ($07F6)= Y register	
TRAP	Turns on or off error interception	TRAP [<line>] <line> - BASIC line number where program execution should continue when an error occures	
TROFF	Turns trace mode off	TROFF	
TRON	Turns trace mode on	TRON	
VER	Show version of this BASIC extension	VER	
VERIFY	Checks stored program against the one in memory	VERIFY "<file>"[,<device>[,<rel_flag>]]	
VOL	Sets sound volume level	VOL 0-8	
WAIT	Waits for a change of memory address	WAIT <address>,<ctrl_value1>[,<ctrl_value2>] <address> - memory location to be monitored (0-65535) <ctrl_value1> - first control value (0-255) <ctrl_value2> - second control value (0-255)	
WINDOW	Create a screen section as fix window.	WINDOWS X1,Y1,X2,Y2	
XOR	8Bit Exclusive OR	XOR X,Y,S XOR 1,1,S -> Result 0 in S XOR 0,1,S -> Result 1 in S XOR 1,0,S -> Result 1 in S XOR 0,0,S -> Result 0 in S	

* D = only direct Mode, P = only program Mode, blank = both (most cases)

Functions

Functions	Description	FORMAT	D/P
¶ (PI)	Returns the value of pi	¶(<dummy>)	
ABS	Returns the magnitude of the numeric value	ABS(<number>)	
ASC	Returns character's ASCII code	ASC(<string>)	
ATN	Returns arctangent	ATN(<number>)	
CHR$	Returns a character in the base of ASCII code	CHR$(<ascii_code>)	
COS	Returns cosine value	COS(<angle>)	
DEC	Converts hexadecimal number to decimal	DEC(<HEXSTRING>) (0000-FFFF)	
ERR$	Returns string describing error condition	ERR$(<err_condition>)	
EXP	Raises constant e to the given power	EXP(<power>)	
FN	Calls user-defined function	FN<fnc_name>(<number>)	
FREE	Returns the amount of available memory	FREE(Dummy)	
HEX$	Converts a decimal number into a hexadecimal one	HEX$(<number>)	
INSTR	Searches for a substring	INSTR(<string_1>,<string_2>[,<start_pos>])	
INT	Extracts the integer portion of a decimal number	INT(<number>)	
JOY	Polls joystick port	JOY(<port>)	
LEFT$	Strips string from the right	LEFT$(<string>,<length>)	
LEN	Returns the number of characters in the string	LEN(<string>)	
LOG	Returns the natural log of the given number	LOG(<number>)	
MID$	Returns a substring	MID$(<string>,<start_pos>,<length>)	
PEEK	Gives contents of memory location (8Bit)	PEEK(<address>)	
POS	Current cursor x position	POS(<dummy>)	
RCLR	Returns color source's current color	RCLR(<color_src>) 0 - background 1 - foreground 2 - multicolor 1 3 - multicolor 2 4 - border	
RDOT	Returns information about the current PC location	RDOT(<info_flag>) 0 - current pixel cursor x position 1 - current pixel cursor y position 2 - color source used at current PC position	

RGR	Returns current graphic mode	RGR(<dummy>)	
RIGHT$	Strips string from the left	RIGHT$(<string>,<length>)	
RLUM	Returns color source's current luminance	RLUM(<color_src>) -> Returns current luminance: 0-7 0 – background 1 – foreground 2 – multicolor 1 3 – multicolor 2 4 – border	
RND	Generates a random number	RND(<seed>)	
SGN	Returns number's sign	SGN(<number>) +1 if <number> is positive 0 if <number> is zero -1 if <number> is negative	
SIN	Returns sine value	SIN(<angle>)	
SPC	Skips over spaces	SPC(<Number of spaces>)	
SQR	Returns the square root	SQR(<number>)	
STR$	Converts number into a string	STR$(<number>)	
TAB	Sets cursor's x position	TAB(<column>)	
TAN	Returns tangent value	TAN(<angle>)	
USR	Executes a machine language program with a parameter	USR(<parameter>)	
VAL	Converts string into a number	VAL(<string>)	

Operators

Operators	Description
+	Addition
-	Subtraction
*****	Multiplication
/	Division
^	raising to a power (exponentiation); ^ = up arrow
=	equal to
<	Less than
>	Greater than
<=	less than or equal to
>=	greater than or equal to
<>	not equal to
><	not equal to
AND	Boolean AND
NOT	Boolean NOT
OR	Boolean OR

Reserved Variables

There are seven variable names which are reserved for use by the computer, and may not be used for another purpose.

Variables	Description
DS	Disk drive's status in numbers
DS$	Disk drive's status in words
EL	Last error line
ER	Stores error condition number (See BASIC Error Messages)
ST	ST is a status variable for input / output. The value depends on the results of the last input/output operation.
TI	Clock value in 1/60 in a seconds
TI$	Current time in format of "HHMMSS"

BASIC Error Messages

These error messages are printed by BASIC. You can also PRINT the messages through the use of the ERR$ function.

Error number	Message	Description
1	TOO MANY FILES	There is a limit of 10 files OPEN at one time.
2	FILE OPEN	An attempt was made to open a file using the number of an already open file.
3	FILE NOT OPEN	The file number specified in an I/O statement must be opened before use.
4	FILE NOT FOUND	No file with that name exists (disk).
5	DEVICE NOT PRESENT	The required I/O device not available.
6	NOT INPUT FILE	An attempt made to GET or INPUT data from a file that was specified as output only.
7	NOT OUTPUT FILE	An attempt made to send data to a file that was specified as input only.
8	MISSING FILE NAME	An OPEN, LOAD, or SAVE to the disk generally requires a file name.
9	ILLEGAL DEVICE NUMBER	An attempt made to use a device improperly (SAVE to the screen, etc.).
10	NEXT WITHOUT FOR	Either loops are nested incorrectly, or there is a variable name in a NEXT statement that does not correspond with one in a FOR.
11	SYNTAX ERROR	A statement is unrecognizable by BASIC. This could be because of missing or extra parenthesis, misspelled keyword, etc.
12	RETURN WITHOUT GOSUB	A RETURN statement encountered when no GOSUB statement was active.
13	OUT OF DATA	A READ statement encountered, without data left unREAD.
14	ILLEGAL QUANTITY	A number used as the argument of a function or statement is outside the allowable range.
15	OVERFLOW	The result of a computation is larger than the largest number allowed (1.701411833E+38).
16	OUT OF MEMORY	Either there is no more room for program and program variables, or here are too many DO, FOR, or GOSUB statements in effect.
17	UNDEF'D STATEMENT	A line number referenced does not exist in the program.

18	BAD SUBSCRIPT	The program tried to reference an element of an array out of the range specified by the DIM statement.
19	REDIM'D ARRAY	An array can only be DIMensioned once. If an array is referenced before that array is DIM'd, an automatic DIM (to 10) is performed.
20	DIVISION BY ZERO	Division by zero is not allowed.
21	ILLEGAL DIRECT	INPUT or GET statements are only allowed within a program.
22	TYPE MISMATCH	This occurs when a number is used in place of a string or vice-versa.
23	STRING TOO LONG	A string can contain up to 255 characters.
24	FILE DATA	Bad data read from a tape.
25	FORMULA TOO COMPLEX	Simplify the expression (break into two parts or use fewer parentheses).
26	CAN'T CONTINUE	The CONT command does not work if the program was not RUN, there was an error, or a line has been edited.
27	UNDEF'D FUNCTION	A user defined function referenced that was never defined.
28	VERIFY	The program on tape or disk does not match the program in memory.
29	LOAD	There was a problem loading. Try again.
30	BREAK	The stop key was hit to halt program execution.
31	CAN'T RESUME	A RESUME statement encountered without TRAP statement in effect.
32	LOOP NOT FOUND	The program has encountered a DO statement and cannot find the corresponding LOOP.
33	LOOP WITHOUT DO	LOOP encountered without a DO statement active.
34	DIRECT MODE ONLY	This command is allowed only in direct mode, not from a program.
35	NO GRAPHICS AREA	A command (DRAW, BOX, etc.) to create graphics encountered before the GRAPHIC command was executed.
36	BAD DISK	An attempt failed to HEADER a disk, because the quick header method (no ID) was attempted on an unformatted disk, or the disk is bad.

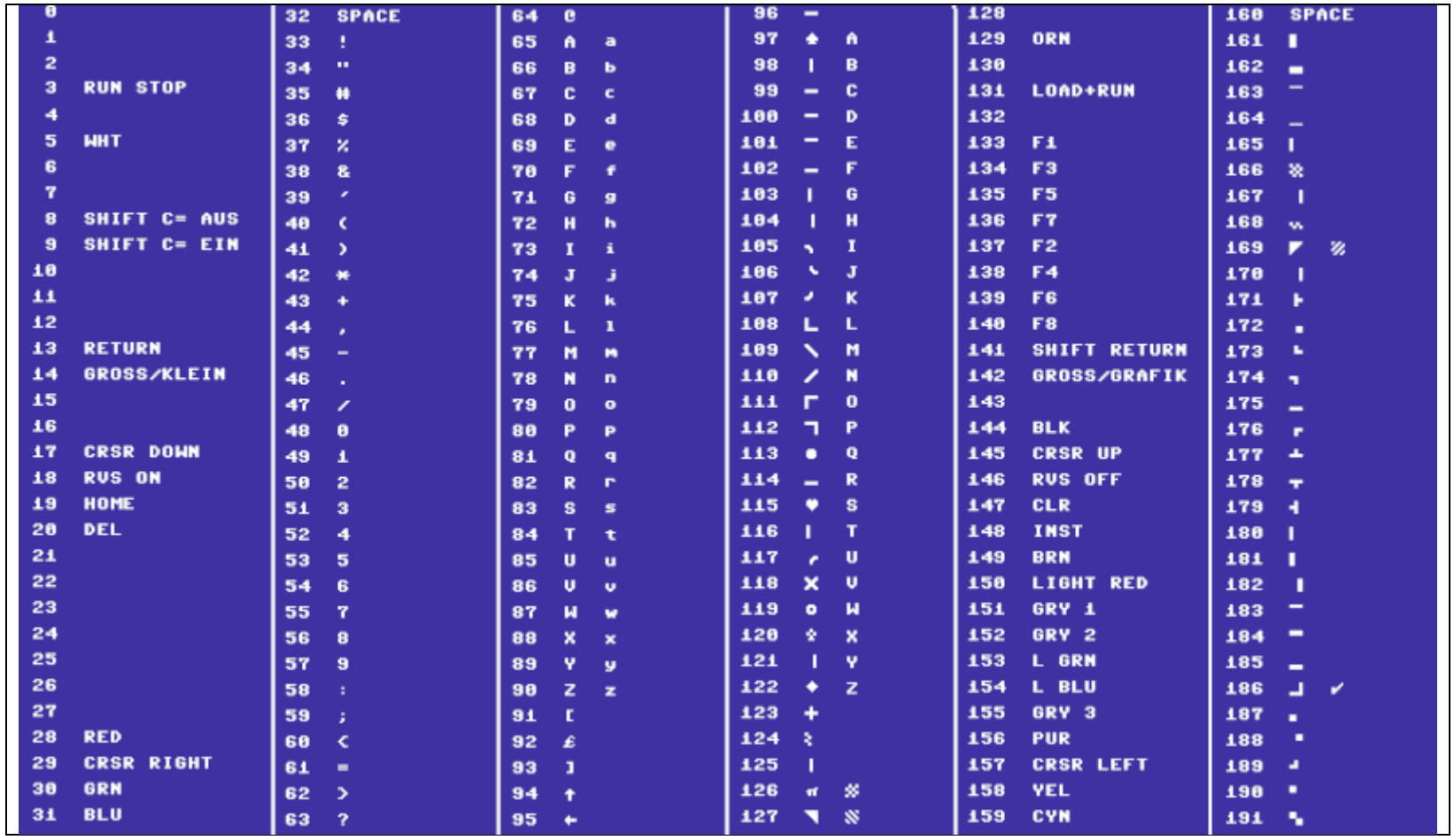
DISK Error Messages

Error number	Message	Description
20	Block header not found.	The disk controller is unable to locate the header of the requested data block. Caused by an illegal sector number, or the header has been destroyed.
21	No sync character.	The disk controller is unable to detect a sync mark on the desired track. Caused by misalignment of the read/write head, no disk is present, or unformatted or improperly seated disk. Can also indicate a hardware failure.
22	Data block not present.	The disk controller has been requested to read or verify a data block that was not properly written. This error message occurs in conjunction with the BLOCK commands and indicates an illegal track and/or sector request.
23	Checksum error in data block.	This error message indicates that there is an error in one or more of the data types. The data has been read into the DOS memory, but the checksum over the data is in error. This message may also indicate grounding problems.
24	Byte decoding error.	The data or header has been read into the DOS memory, but a hardware error has been created due to an invalid bit pattern in the data byte. This message may also indicate grounding problems.
25	Write-verify error.	This message is generated if the controller detects a mismatch between the written data and the data in the DOS memory.
26	WRITE PROTECT ON	This message is generated when the controller has been requested to write a data block while the write protect switch is depressed. Typically, this is caused by using a disk with a write protect tab over the notch.
27	Checksum error in header.	The controller has detected an error in the header of the requested data block. The block has not been read into the DOS memory. This message may also indicate grounding problems.
28	Too long data block.	The controller attempts to detect the sync mark of the next header after writing a data block. If the sync mark does not appear within a pre-determined time, the error message is generated. The error is caused by a bad disk format (the data extends into the next block), or by hardware failure.

29	DISK ID MISMATCH	This message is generated when the controller has been requested to access a disk which has not been initialized. The message can also occur if a disk has a bad header.
30	Error in general syntax.	The DOS cannot interpret the command sent to the command channel.
31	Invalid command.	The DOS does not recognize the command. The command must start in the first position.
32	Invalid command.	The command sent is longer than 58 characters.
33	Invalid file name.	Pattern matching is invalidly used in the OPEN or SAVE command.
34	SYNTAX ERROR	No file given. The file name was left out of a command or the DOS does not recognize it as such. Typically, a colon (:) has been left out of the command.
39	Invalid command.	This error may result if the command sent to command channel (secondary address 15) is unrecognized by the DOS.
50	RECORD NOT PRESENT	Result of disk reading past the last record through INPUT#, or GET# commands. This message will also occur after positioning to a record beyond end of file in a relative file. If the intent is to expand the file by adding the new record (with a PRINT# command), the error message may be ignored. INPUT or GET should not be attempted after this error is detected without first repositioning.
51	OVERFLOW IN RECORD	PRINT# statement exceeds record boundary. Information is truncated. Since the carriage return which is sent as a record terminator is counted in the record size, this message will occur if the total characters in the record (including the final carriage return) exceeds the defined size.
52	FILE TOO LARGE	Record position within a relative file indicates that disk overflow will result.
60	WRITE FILE OPEN	This message is generated when a write file that has not been closed is being opened for reading.
61	FILE NOT OPEN	This message is generated when a file is being accessed that has not been opened in the DOS. Sometimes, in this case, a message is not generated; the request simply ignored.
62	FILE NOT FOUND	The requested file does not exist on the indicated drive.
63	FILE EXISTS	The file name of the file being created already exists on the disk.
64	FILE TYPE MISMATCH	The file type does not match the file type in the directory entry for the requested file.

65	NO BLOCK	This message occurs in conjunction with the B-A command. It indicates that the block to be allocated has been previously allocated. The parameters indicate the track and sector available with the next highest number. If the parameters are zero (0), then all blocks higher in number are in use.
66	ILLEGAL TRACK AND SECTOR	The DOS has attempted to access a track or block which does not exist in the format being used. This may indicate a problem reading the pointer to the next block.
67	ILLEGAL SYSTEM T OR S	This special error message indicates an illegal system track or sector.
70	NO CHANNEL	The requested channel is not available, or all channels are in use. A maximum of five sequential files may be opened at one time to the DOS. Direct access channels may have six opened files.
71	DIRECTORY ERROR	The BAM (Block Availability Map) does not match the internal count. There is a problem in the BAM allocation or the BAM has been overwritten in DOS memory. To correct this problem reinitialize the disk to restore the BAM in memory. Some active files may be terminated by the corrective action.
72	DISK FULL	Either the blocks on the disk are used or the directory is at its entry limit. DISK FULL is sent when two blocks are available on the 1541 to allow the current file to be closed.
73	DOS MISMATCH	DOS 1 and 2 are read compatible but not write compatible. Disks may be interchangeably read with either DOS, but a disk formatted on one version cannot be written upon with the other version because the format is different. This error is displayed whenever an attempt is made to write upon a disk which has been formatted in a non-compatible format. (A utility routine is available to assist in converting from one format to another.) This message may also appear after power up.

PETascii codes



Musical note table

A - sound register value 7 use the 7 as a second number after the SOUND command - SOUND 1,7,30.

NOTE	REGISTER (PAL)	REGISTER (NTSC)	FREQUENCY (Hz)	NOTE	REGISTER (PAL)	REGISTER (NTSC)	FREQUENCY (Hz)
A	7	7	110.0	#F	873	873	740.0
#A	64	64	116.6	G	881	881	784.0
H	118	118	123.5	#G	889	889	830.7
C	169	169	130.9	A	897	897	880.0
#C	217	217	138.6	#A	904	904	932.4
D	262	262	146.9	H	911	911	987.8
#D	305	305	155.6	C	917	917	1046.6
E	345	345	164.9	#C	923	923	1108.8
F	383	383	174.7	D	929	929	1174.7
#F	419	419	185.0	#D	934	934	1244.6
G	453	453	196.0	E	939	939	1318.6
#G	485	485	207.7	F	944	944	1397.0
A	516	516	220.0	#F	948	948	1480.0
#A	544	544	233.1	G	953	953	1568.0
H	571	571	247.0	#G	957	957	1661.3
C	596	597	261.7	A	960	960	1760.0
#C	620	621	277.2	#A	964	964	1864.7
D	643	643	293.7	H	967	967	1975.6
#D	664	665	311.2	C	971	971	2093.0
E	685	685	329.7	#C	974	974	2217.5
F	704	704	349.3	D	976	976	2349.4
#F	722	722	370.0	#D	979	979	2489.1
G	739	739	392.0	E	982	982	2637.1
#G	755	755	415.4	F	984	984	2793.9
A	770	770	440.0	#F	986	986	2960.0
#A	784	784	466.2	G	988	988	3136.0
H	798	798	493.9	F	864	864	698.5
C	810	810	523.3	E	854	854	659.3
#C	822	822	554.4	#D	844	844	622.3
D	834	834	587.4				

Special particularities of BASIC 4.5

After BASIC 4.5 has started, it is checked whether a program with the name "BOOT" is on drive 0, unit 8. If so, it will be loaded and executed.

If a RTC module DS12C887 with base address $DE00 is present, then BASIC 4.5 will set TI and TI$ automatically after each start process.

The “IF THEN” problem
Please use a colon (:) after each THEN to prevent a SYNTAX ERROR. That means:
NO: -> IF A=1 THEN CLS This occurs a SYNTAX ERROR
YES: -> IF A=1 THEN: CLS That’s working perfect.
Usually a colon can always be placed after a THEN

TEDMON

After the start is shown this message:

```
    PC SR AC XR YR SP
; 0004 30 00 00 00 F6
```

The first line names the CPU registers and the second shows their current content. The abbreviations in the register line mean:

- PC: **P**rogram **C**ounter; memory address of the next assembler command
- SR: Content of **S**tatus **R**egister
- AC: Content of **AC**cumulator
- XR: Content of **X**-index **R**egister
- YR: Content of **Y**-index **R**egister
- SP: Content of **S**tack **P**ointer
- IRQ: Interrupt vector

The following commands can used:

- **A** - **A**ssemble a mnemonics line into machine code.
  - **A** <adrese> <command> [<operand>]

- **C** - **C**ompare two memory aeras and displays the difference.
  - **C** <start adress> <end adress> <start adress for comparing>

- **D** - **D**isassemble a machine code line into mnemonics.
  - **D** [<start adress > [<end adress>]]

- **F** - **F**ill up a memory aera with the given byte.
  - **F** <start adress> <end adress> <Byte>

- **G** - **G**o to the memory adress, also start a machine code program at the inputed memory adress.
  - **G** <adress>

- **H** - **H**unt a memory aera - Durchsucht Speicherbereich nach einen bestimmten Wert und zeigte alle gefundenen Speicherstellen an
  - **H** <start adress> <end adresse> <datas> (datas are hexadecimal numbers seperated with empty spaces and strings seperated with the prefix apostrophe (').)

- **L** - **L**oad a file from disk or datasette into the memory.
  - **L** "<filename>",<device number ($1-$F)>,<load memory adress at C128>

- **M** - **M**emory is showing in hexadecimal numbers and values.
  - **M** [<start adress> [<end adress>]] (by using this command without adresses the first 12 lines are shown.)

- **R** - **R**egisters is shown again.
- **S** - **S**ave the inputed memory aera into a file on disk or datasette.
  - **S** "<file iname>",<device number>,<start adress>,<end adress+1>

- **T** - **T**ransfer) or copy a memory aera into another.
  - **T** <start adress> <end adress> <destination adress>

- **V** - **V**erify a saved file on disk or datasette with the memory aera.
  - **V** "<file name>",<device number ($1-$F)>,<start adress>

- **X** - e**X**it TEDMON into BASIC direct mode.

- **>** - Modify one until eight bytes in a memory adress (after M command).
  - **>** <adress> <byte1> <byte2> ... <byte8>

- **.** - Works same like the A command
- **;** - Change the register content (after R command)

TEDMON hasn't a input prompt! Only a blinking cursor is shown that the machine code monitor is ready.
If a wrong input is done (unknown command) a question remark **?** appears.

ESCAPE CODES

The ESCAPE Codes extend the functionality of the BASIC Editor

Cancel quote and insert mode	ESC O
Cancel started Esc code	ESC X
Erase to end of current line	ESC Q
Erase to start of current line	ESC P
Move to start of current line	ESC J
Move to end of current line	ESC K
Enable auto-insert mode	ESC A
Disable auto insert mode	ESC C
Delete current line	ESC D
Insert line	ESC I
Enable scrolling	ESC L
Disable scrolling	ESC M
Scroll up	ESC V
Scroll down	ESC W
(It is usefull with the WINDOW command. See page 9)	
Set bottom of screen window	ESC B
Set top of screen window	ESC T
Set window to full screen minus 1 and clear	ESC R
Set window to full screenand clear screen	ESC N

BASIC 4.5 Tokenlist

All BASIC 4.5 Token are 2 Byte tokens. The "Mastertoken" is 7F.

Keyword	Token	Keyword	Token
AT	7F 01	**BEGINBLOCK**	7F 0D
CLS	7F 02	**ENDBLOCK**	7F 0E
INFO	7F 03	**RESTORE**	7F 0F
VERSION	7F 04	**RTRIM**	7F 10
FRAME	7F 05	**LTRIM**	7F 11
CATALOG	7F 06	**XOR**	7F 12
MEMORY	7F 07	**'**	7F 13
STOP	7F 08	**DELAY**	7F 14
WINDOW	7F 09	**BEEP**	7F 15
DLRUN	7F 0A	**CALL**	7F 16
FIND	7F 0B	**JUMP**	7F 17
FILES	7F 0C	**[for future use]**	7F 18

Room for additions

Use this page for your own notice

Keyword	Description