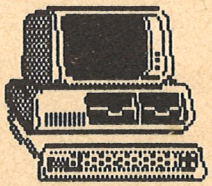
PRINT TECHNIK

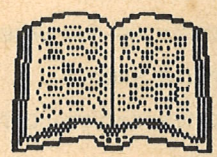
Help PC-128



Plus C-64

by Peter Schwarz

Users Manual



A-1060 Vienna / From June 1st 1986
Stumpergasse 34 / Tel. 5973423
D-8000 München /
Nikolaistraße 2 / Tel. 089-368197

TABLE OF CONTENTS

GENERAL

HELP 128 PLUS C 64 is a modul to make programming and checking of own programs easier. It will be delivered as a Cartridge-modul. This modul must be inserted in the CARTRIDGE EXPANSIONS SLOT after you have switched off the Commodore. After inserting and turning on again the Computer you will see on the monitor the normaly SWITCH ON picture and the statement PRINT-TECHNIK HELP 128 PLUS 64, or * PRINT-TECHNIK HELP C64 PLUS *

HELP 128 PLUS C 64 requires no BASIC area.

For the C 128 only, in BANK 0 the area from $1600 - 1690 is required for HELP 128 PLUS C 64 and may not be destroyed by the user. Otherwise HELP 128 PLUS C 64 requires only areas which will be also used from BASIC by similar commands.

Monitor and Disassembler also available for Floppy.

For the C 128 only: TRACE and SINGLESTEP also with variables.

For the C 128 only: RENUMBER and COMPACTER 10 times faster.

For the C 128 only: UNCOMPACTER.

All HELP commands must only be entered in DIREKT MODE (not in a programm) and have to begin in column 1. For the execution of commands only RAM-MEMORY LOCATIONS of ZERO PAGE and STACK PAGE (0110 - 0135 Hexa) are used. There are no changes in memory locations which may prevent continuing the program by CONT.

HELP 128 PLUS C 64 consist of 4 parts:

1. BASIC-additional commands for programming and execution:

   22 commands: APPEND, BASIC-RENEW, COMPACT, UNCOMPACT, DELETE, FIND, GENLINE, HELP, SINGLE STEP, TRACE, END of SINGLE STEP and TRACE, KILL, RENUMBER, VARIABLE, MATRIZ, PAGE-LIST, HEXA to DECIMAL, DECIMALE to HEXA, CHECKING for UNDEF'D STATEMENT, opening FILE with CMD, END of CMD, PRINTFRE(0), PRINTFRE(1), DEVICE-modify.

2. DISASSEMBLER-MONITOR:

   13 commands for programming, checking, listing and for execution of machine language programs in a C-64 memory and a FLOPPY-DISK memory. It represents a MONITOR in combination with a DISASSEMBLER.

3. DOS-commands:

   8 commands for Floppy-Disk. Disk commands are simplified.

4. 2-PASS ASSEMBLER for 6500-6515 MICROPROCESSOR

The operation of each command is explained on the following pages in detail.

HELP 128 PLUS C 64 restarts automatically after each reset. However, you ca. terminate it by the command: #K. If you want to start it again without turning the C-64 off, just typ SYS 33000. If you work with the C 128, you must type: BANK8:SYS32768 and the HELP 128 PLUS C 64 will be restart.

A P P E N D

COMMAND-CODE: #A

SYNTAX: #A "NAME", D (similar to LOAD; D = DEVICE)

If DEVICE is omitted, TAPE is assumed. NAME may only be missing by TAPE. Then the next program will be loaded. By means of this command a program from tape or floppy will be added behind a program existing in the memory. Line numbers of the new program may be less than or equal to those in the old program. Also length of the single programs may differ.

The joined programs can be RENUMBERed with #R. NOTE: At the commands GOTO, GOSUB, THEN or RUN, if they have equal line numbers, always the first occuring line number is used. It is recommended - before joining two or more programs - to RENUMBER the single programs that there is no double line number and that the line number of the new program is higher than the number existing in the memory.

RESTRICTION: Machine language programs which are behind a program in the memory, are deleted by the added program.

H E L P

COMMAND-CODE: #H

SYNTAX: #H

If the execution of a program had been interrupted, this command allows listing of the last executed command line.
When interrupting is caused by a program error (ERROR) the error-causing part is displayed in REVERS form. This must not be the real error because a program error can cause later on an interruption and only this interruption address is known.

This command shall be used immediately after termination of a program because entering another command may change memory locations which are necessary for the operation and this command can no longer be executed.

F I N D

COMMAND-CODE: #F

SYNTAX:	#F COMMAND	(lists all lines including this command)
	#F VARIABLE	(lists all lines including this variable name)
	#F "STRING	(lists all lines including this string)

This command gives you the possibility to list all commands, variables, figures or strings in program.

Upon searching for a VARIABLE with a single letter (eg. A), all lines with a two-letter variable having the same first letter will be listed also (eg. a2, az)

Upon searching for a STRING, it is not necessary that it must be at the beginning of quotation mark. Entering of #F" (without string) lists all lines where a string occurs.

Upon searching for a NUMBER (eg. 122), all lines with a figure including this part will be listed (eg. 21226).

Upon searching for a COMMAND, this command may also be entered in an abbreviated form (eg. OP for OPEN). It is also possible to use COMMANDS with NUMBERS, eg. GOSUB 500 or with VARIABLES, eg. IF A. In this case it is important to enter in the same way as in a program, eg. GOSUB 500 if no space had been left or GOSUB 500 with space in program.

Listing of all displayed lines can be slowed down by the CTRL key and locked by the SHIFT key (as long as you press it). Termination by the STOP key.

These lines can be printed with a printer by using the : "CMD " command (like a list-printing).

D E L E T E

COMMAND-CODE: #D

SYNTAX:	#D 100	(erases line 100)
	#D -100	(erases up to line 100)
	#D 100-	(erases all beginning at line 100)
	#D 100-200	(erases all from line 100 to line 200)
	#D	without operand is treated as SYNTAX ERROR to prevent disturbing the program unintentionally.

With this command a part of the program can be deleted without entering each line number. A machine language program behind this program is NOT DISTURBED by this command.

After execution of this command a continuation of a program with CONT is no longer possible.

G E N L I N E

COMMAND-CODE: #G

SYNTAX f. C 64:	#G A,B	A = initial line number B = steps between linenumbers
	#G	without A and B: 100, 10 is asumed.
SYNTAX f. C 128:	#G A	A= steps between line number
Repetition of	#G	End of GENLINE

After entering this command each typing of the RETURN-key prints a new line number. The line number is incremented automatically by the value B. This command is useful for writing a program.

With the C 64, this program is terminated by deleting the LINE NUMBER by DEL and pressing RETURN. The Program can also be terminated by pressing SHIFT-RETURN (without deleting of the line number).

With the C128, this program is termineted, if you repeat #G.

The initial line number must be in the range 0 to 63999. The increment of the line numbers must be in the range 1 to 255. If you put in other numbers, you see: ILLEGAL QUANTITY ERROR. If you reach a value higher than 63999 then 64000 is subtracted from this value and the rest is displayed as present line number.

TRACE

COMMAND-CODE: #T

SYNTAX: #T (C 64 and C 128)
SYNTAX: #T,A,B,C (for C 128 only. Shows the actual values of 3 variables)

This command causes the execution of a program line by line. Execution stops for a short time after executing one line. Then it continues at the next line. If the SHIFT key is pressed during execution, the program runs very slowly.

When starting a BASIC-program with RUN, GOTO, GOSUB or CONT, at the top screen line the line number of the next line which will be executed appears. By execution of this line, this line number will be moved to the left side and in front of this line number the line number of the next line which will be executed is displayed. The last 6 line numbers are displayed.

The content of the next program line which will be executed is displayed in the second and the third screen line.

The first three screen lines are deleted each time before a new command will be executed. INPUT-commands are also a part of an entry if these commands are in the first 3 lines. Nevertheless to continue the program without errors it is possible to put the cursor on the fourth line and then do the entry.

The program can be interrupted every time with the STOP key and continued with CONT.

This command is present until it is terminated by the command #E = END TRACE and STEP.

The six line numbers are stored in the memory locations from $0122 - $0130. These memory locations should not be modified by POKE by execution of this command.

SINGLE STEP

COMMAND-CODE: #S

SYNTAX: #S (C 64 and C 128)
SYNTAX: #S,A,B,C (For the C 128 only. Else like TRACE)

This command causes a Basic-program to be executed line by line. Execution stops until one of the two SHIFT KEYS is depressed shortly. If one SHIFT KEY is held down, Basic-program runs slowly (similar to TRACE).

When starting execution with RUN, GOSUB or CONT or GOTO, at the top screen line the line number of the next line which will be executed appears. By execution of this line, this line number will be moved to the left side and in front of this line number the line number which will be executed is displayed. The last 6 line numbers are displayed.

The content of the next programm line which will be executed is displayed in the second and the third screen line.

The first three screen lines are deleted each time before a new command will be executed. INPUT-commands are also a part of an entry if these commands are in the first 3 lines. Nevertheless to continue the program without errors it is possible to put the cursor on the fourth line and then do the entry.

The program can be interrupted all the time with the STOP key and continued with CONT.

This command is present until it is terminated by the command #E = END TRACE and STEP.

The six line numbers are stored in the memory locations from $0122 - $0130. These memory locations should not be modified by POKE by execution of this command.

END of TRACE and STEP

COMMAND-CODE: #E

SYNTAX: #E
This command changes the execution of a Basic-program back to the NORMAL-MODE. For switching from SINGLE-STEP to TRACE or vice versa the command #E is not necessary.

RENUMBER

COMMAND-CODE: #R

SYNTAX: #R A,B — A = initial line number, B = increment of line numbers

#R — without A and B: 100, 10 is assumed.

This program renumbers a program stored in memory. The programm checks automatically for UNDEF'D STATEMENT . At the same time all GOTO, GOSUB, IF THEN, ELSE, COLLISION, TRAP, RESUME and RUN statements followed by one or at ON statement by more line numbers, these line numbers are also changed. If the programm find this error you should correct it, because if a nonexisting line number appears it will be numbered to 65535.

With this command a program can become longer or shorter depending on the initial line number and the icrement.

The initial line number has to be in the range from 0 to 63999. The increment of the line numbers has to be in the range from 1 to 255. By the entry of other numbers the program will be interrupted with: ILLEGAL QUANTITY. NOTE: If there is a line number greater than 63999, from this figure 64000 are subtracted assumed as new line number. The order of these program lines, however, is the same.

A program can be renumbered repeatly without being destroyed. A machine language program after the Basic program is not altered. If the program find a line number by renumbering which is greater than 63999, execution is interrupted and you see: SYNTAX ERROR.

VARIABLES DUMP

COMMAND-CODE: #V

SYNTAX: #V

With this command all non arrayed VARIABLES used in a program can be listed with their current value after terminating the execution.

Listing is made in the sequence of the activation of the variables. All variables are printed in such a way that their values can be listed, edited to new values and with the CONT command given to continue the program with the new values.

Listing of VARIABLES can be slowed down by CTRL key and hold on with SHIFT key (as long as you press it). You can interrupt by stop key.

It is also possible to print the VARIABLES with a printer like a normal listing by ((CMD).

ARRAY DUMP

COMMAND-CODE: #M

SYNTAX: #M

With this command all ONE DIMENSIONAL or MORE DIMENSIONAL VARIABLES used in program can be listed with the current values after terminating the execution.

Listing is made in the sequence of the activation of the variables. All variables are printed in such a way that their values can be listed, edited to new values and with the CONT command given to continue the program with the new values.

Listing of the VARIABLES can be slowed down by CRTL key and hold on with SHIFT key (as long as you press it). You can interrupt by STOP key.

It is also possible to print the VARIABLES with a printer like a normal listing by ((CMD).

PAGE LIST

COMMAND-CODE: #L

SYNTAX: #L n n = line number

#L without line number = listing from begin

Listing begins at the first line number of the program and lists the contents page by page (25 lines) on the screen. To list the next page you have to press the RETURN key.
If you press the ARROW UP KEY: ↑ the page before is listed.

This program can only be terminated by pressing the STOP key. The cursor is always situated in the left bottom corner and the listed program on the screen will not be scrolled upwards.

As this command is only appointed to list on the screen an existing CMD will be terminated before execution and not opened again.

CONVERSION from HEXA to DECIMAL

COMMAND-CODE: !#

SYNTAX: !#HHHH

A HEXADECIMAL NUMBER up to 4 digits is converted to a DECIMAL NUMBER. For the values 10 to 15 the letters A to F have to be used.

CONVERSION from DECIMAL to HEXA

COMMAND-CODE: !$

SYNTAX: !$DDDDD

A DECIMAL NUMBER (positive from 0 to 65535) is converted to a HEXADECIMAL NUMBER up to 4 digits. For the values 10 to 15 letters A to F are used. With this command also variables can be used.

BASIC RENEW

COMMAND-CODE: #B

SYNTAX: #B

This command can be used threefold:

1. A BASIC program deleted by NEW is activated with this command.
2. A BASIC program is set free from a machine language program which is appended or a progr. part which has been appended by OVERLAY mistakenly Program-end Pointer is set new.
3. If you have loaded a BASIC program and after loading no CURSOR appears and only with STOP-RESTORE the CURSOR appears, this command should be used to set new the program Pointer.

This command is not allowed to be used if no BASIC program is in memory because it is possible that program lines with undefined content can occur.

Opening of FILES and CMD

COMMAND-CODE: (

SYNTAX: (d d = DEVICE NUMBER (4 - 31)
(without d: DEVICE 4 is assumed

This command opens an OUTPUT FILE with the FILE NUMBER 255. If this FILE is already opened it does not lead up a FILE OPEN ERROR with CLOSING OF ALL FILES but the existing open FILE will be used for the CMD. By this command a SECONDARY ADDRESS is not possible.

This command does the following command parts:
1. OPEN 255,d d = DEVICE
2. CMD 255

CMD and CLOSING OF FILES

COMMAND-CODE:)
SYNTAX:)
This command terminates a CMD and closes the FILE 255.

MODIFYING OF A DEVICENUMBER for DOS

COMMAND-CODE:£

SYNTAX: £ d d = DEVICE NUMBER (4 - 31)

With this command the DEVICE NUMBER for DOS and ASSEMBLER can be modified. Every time you switch on the HELP 128 PLUS C 64 the DEVICE NUMBER will be 8.

KILL

COMMAND-CODE: #K

SYNTAX: #K

With this command HELP 128 PLUS C 64 is switched off. HELP 128 PLUS C 64 is then completely turned off and there is no interruption for other programs with the same SYNTAX.

COMPACTOR

COMMAND-CODE: #C

SYNTAX: #C d d = DENSITY (1 - 240)

#C without d: DENSITY 240 is assumed.

With this command a BASIC program is condensed. All unnecessary SPACES and all REM are removed. Lines are joined till the length (density) wanted is achieved except if they are jumped-on lines.

Before the program is compacted, it is checked for UNDEF'D STATEMENT. If this error occours, execution is interrupted and the compacting stopped.

After compacting the program is RENUMBERed by 1,1 to get an absolut density.

COMPACTED programs run faster because the time for finding a line is shorter for jumping commands. Unnecessary SPACES and REM cost time. A once compacted program is very difficult to disentrangle and to modify. A compacted program should only be used for the execution of a program and not for the test phase. Keep always a copy of the original uncompacted, in case you want to change a line.

UNCOMPACTOR

COMMAND-CODE: #C0
SYNTAX: #C0

This program works with the C 128 only! Programmes compacted with the

command #C, will make uncompacted.

CHECKING for UNDEF'D STATEMENT

COMMAND-CODE: #U

SYNTAX: #U

The BASIC program existing in memory is checked for UNDEF'D STATEMENT. If such an error is found you see the line number of the program line in this FORM:
UNDEF'D STATEMENT ERROR in 100 (if this error has been found in line 100).

FREE MEMORY LOCATIONS

COMMAND-CODE f. C 64: *

SYNTAX: *

COMMANDCODE for C 128: * or *0 (free memory for program)
*1 (free memory for variable)
SYNTAX f. C 128: *0 or *1

This command replaces the command: PRINTFRE (0) of the C 64, also the PRINTFREE(0) and the PRINTFREE(1) command of the C 128

DISASSEMBLER MONITOR

COMMAND-KEY:]

With this key you are starting a MONITOR program where the BASIC-EDITOR is disabled. This program services 13 commands which are described on the followingpages in detail.

MONITOR means: The address of a memory location is often shown by an 16 BIT address direction and also by this microprocessor and by this MONITOR in HEXADECIMAL (short: HEXA) which has up to 4 digits. These contents (DATA) are shown by an 8 BIT data direction and also by this microprocessor and by this MONITOR in HEXA which has up to 2 digits. By this MONITOR the address will be displayed on the left side. The content (Data) is displayed two spaces right of them.

For indication and input of a HEXA value the digits 0 - 9 and for the value 10 - 15 the letters A - F are used.

The DISASSEMBLER-MONITOR allows 3 MODES (switch key) which are in a defined position when starting this program each time. For MODE you can think of a switch key which remains in a position until it is switched over.

MODE 1: MODIFYING of ADDRESS or DATA

Each time when starting the DISASSEMBLER-MONITOR the ADDRESS MODIFY MODE is on. Because the same keys (0 - 9 and A - F) are used for changing of ADDRESS and DATA, it is necessary to determinate, what has to be changed.

MODIFYING of ADDRESS: COMMAND-CODE: +

After typing this command key the keys 0 - 9 and A - F modify the

address in the following way: The entered value is written to the fourth digit, all other values are shifted left one position and the highest value is eliminated. The CONTENTS (DATA) of this new address is printed.

MODIFYING of DATA: COMMAND-CODE: /

After this command key has been depressed, the keys 0 - 9 and A - F modify the address contents (DATA) in the following way. The entered value is shifted to the second digit and the former second value is shifted to the first digit. The former first digit value is eliminated. The entered value is immediately read out and displayed again. If at this address a ROM exists there, the input causes no operation. So the MONITOR can also be used for a ROM/RAM checker.

MODE 2: OUTPUT at SCREEN or PRINTER

This MODE is only valid for the DISASSEMBLER, because for the MONITOR only the screen is used (With MONITOR the contents of the selected memory locations is permanently read out and displayed. This is useful to recognize a TIMER REGISTER or similar memory locations). At each call of the DISASSEMBLER MONITOR the SCREEN MODE is turned on.

OUTPUT to SCREEN DISPLAY: COMMAND-CODE:]

After pressing this command key the DISASSEMBLING output switched to the screen.

OUTPUT to PRINTER: COMMAND-CODE: [

After pressing this command key the DISASSEMBLING output is switched to DEVICE 4 for the printer. It is not necessary to open a FILE.

MODE 3: DISPLAY of COMPUTER MEMORY or DISK MEMORY

BY means of this DISASSEMBLER-MONITOR it is possible to list the memory locations of a connected FLOPPY-DISK (ROM and RAM). Upon each starting of the DISASSEMBLER-MONITOR, program is switched to COMPUTER memory.

COMPUTER MEMORY: COMMAND-CODE: >

After pressing this command key program is turned over to COMPUTER memory and the contents of the COMPUTER memory location is displayed.

DISK MEMORY: COMMAND-CODE: <

After pressing this command key program is turned over to DISK memory and the contents of the DISK memory location is displayed. In this status memory content is only read out for one time in opposition to COMPUTER MEMORY so that the DATABUS is not obstructed constantly.

INCREMENT ADDRESS: COMMAND-CODE: RETURN-KEYE

The address is incremented by one. The new address and its contend overwrite the old line

DECREMENT ADDRESS: COMMAND-CODE: ↑

The address is decremented by one. The new address and its content

overwrite the old line.

RUN: COMMAND KEY: *

With this key a machine language program with the desired address is started. With RTS in a machine language program you jump in the MONITOR again. If DISK MEMORY MODE is on, with this command also a programm in the FLOPPY-DISK can be started. Here MONITOR is not obstructed.

BANKSWITCHING

COMMAND-CODE f. C 128:	,	(comma) or . (dot)
SYNTAX f. C 128:	,	(BANK -1)
	.	(BANK +1)
COMMAND-CODE f. C 64 :	.	(dot
SYNTAX f. C 64:		. (BANK +1) repeat for BANK -1

DISASSEMBLER 1 COMMAND: COMMAND-CODE: -

Typing this command key causes the disassembly of one command starting with the selected address. If PRINT-OUT-MODE is selected, this output is directed to DEVICE 4. If this is no command-code, *** is displayed. All displayed addresses and datas are HEXA-DECIMAL and have not put ahead a $-character for easier readability.

CONTINUED DISASSEMBLER: COMMAND-CODE: SPACE-KEY

Disassembly starts after typing this command key at the selected address until STOP key is depressed. Printing can be slowed down with CTRL as with listing and it can be stopped with a SHIFT key (as long as you press it). If PRINTER-OUTPUT-MODE is selected, this output is directed to DEVICE 4.

TRANSFER (for C 64 only): COMMAND-CODE: @

Only possible in the COMPUTER memory (also in DISK-MODE). With this command programs or datas can be moved from one memory location to another. The contents of the moved datas is the same. Length of the moving program is without LIMIT. Before the command key is allowed to be pressed the following entries have to be made:

1. START ADDRESS of the moving program to the memory locations: 0022 low-part and 0023 high-part
2. END ADDRESS + 1 of the moving program to the memory locations: 0024 low-part and 0025 high-part
3. BEGINNING ADDRESS - where to move the PROGRAM, is to be selected in MONITOR and then typing the command key.

EXAMPLE: The BASIC-ROM (A000 - BFFF) shall be moved up to the RAM memory (starting at 2000).

1. Typing the COMMAND KEY +	(ADDRESS-MODE)
2. Put the ADDRESS to 0022	(Entry: 0 0 2 2)
3. Typing the COMMAND KEY /	(DATA-MODE)
4. Change the contents at 0022 to 00	(Entry: 0 0)
5. Press RETURN key	(INCREMENT to 0023)
6. Change contents of 0023 to A0	(Entry: A 0)
7. Press RETURN key	(INCREMENT to 0024)
8. Change contents of 0024 to 00	(Entry: 0 0 end address + 1 = C000)
9. Press RETURN key	(INCREMENT to 0025)

```
10.Change contents of 0025 to C0          (Entry: C 0)
11.Press COMMAND KEY +                    (ADDRESS-MODE)
12.Put ADDRESS to 2000                    (Entry: 2 0 0 0)
13.Press command key                      (Transfer follows)
```

RETURN to BASIC: COMMAND-CODE: =

DISASSEMBLER-MONITOR program is terminated. An interrupted BASIC program can be continued by CONT after using the DISASSEMBLER-MONITOR.

DOS SUPPORT

This part of program shortens and simplifies the handling of DISK COMMANDS in DIRECT MODE. By HELP 128 PLUS C 64 is preconfigured a default value of 8 for the device number. The following commands are available:

LOAD PROGRAM with relocating: COMMAND KEY: /

A program name must exist after this command key. (It can also be only a part of the name, but it has to end with *). Between command code and program name no quote mark or space may be inserted.

LOAD PROGRAM without relocating: COMMAND KEY: %

Input syntax like LOAD PROGRAM with relocating. The program is not loaded according to the Basic-begin-pointer, but as the disk program-begin-pointer.

LOAD PROGRAM with relocing + RUN: COMMAND KEY: ↑

Input syntax like LOAD PROGRAM. After loading, program starts automatically.

SAVE PROGRAM: COMMAND KEY:←

Input syntax like LOAD PROGRAM. With a dual FLOPPY it is important to enter the DRIVE number before the program name in the following form: 0:NAME for DRIVE 0 or 1:NAME for DRIVE 1. If a STATUS-ERROR occurs during saving, saving is stopped and the DISKSTATUS is listed. If it is a FILE EXISTS ERROR you are asked: OVERWRITE YES/NO. The program on DISKETTE is deleted by pressing Y and the new program is recordet. If the same error occurs once more, the disk should be tested for WRITEPROTECTION. If you enter N nothing is deleted and new recordet.

After termination of saving, DISKSTATUS is displayed.

VERIFY PROGRAM: COMMAND KEY: <

Compare the contents of a BASIC programfile on disk with the program currently in memory.

COMMAND CHANNEL 15 COMMANDS: COMMAND KEY: @

The command following this command key is to enter without comma and quote mark. eg.:

```
@N0:NAME,ID                  = a NEW of a diskette with ID
@I0                          = INITIALIZE DRIVE 0
@S0:NAME                     = delete a program
@B-R:2 0 5 2                 = BLOCK READ to CHANNEL 2 from DRIVE 0
                               TRACK 5 SECTOR 2
@C1:NAME=0:NAME              = COPY of a program
@D1=0                        = DUPLICATE from DRIVE 0 to 1
```

M-command cannot be used with this command. These commands are not longer necessary because of the DISASSEMBLER-MONITOR.

TABLE OF CONTENTS: COMMAND KEY: $

With this command key the table of contents (DIRECTORY) is listed on the screen. If you have a dual floppy you can select with $0 or $1 which drive you want. If you enter CMD before this command, the directory will be listed on the printer.

A program residing in memory will NOT be deleted.

DISK-STATUS INPUT: COMMAND KEY: @ without text

Is this command key (without text) pressed, DISK-STATUS is displayed on the screen, a CMD is closed.

GET CHANNEL 15: COMMAND KEY: > without text

One single character is displayed from the COMMAND CHANNEL in HEXADECIMAL. It is used after the command M-R. NOTE: After M-R no INPUT with the command key @ is allowed because it transmits no END CHARACTER.

ASSEMBLER

COMMAND-CODE: [

HELP 128 PLUS C 64 ASSEMBLER only operates with a FLOPPY-DISK!

HELP 128 PLUS C 64 ASSEMBLER is a 2 PASS-ASSEMBLER and can be called up constantly and uses for its execution no aid programs. For making an ASSEMBLER-SOURCE-PROGRAM you need no own EDITOR because the ASSEMBLER translates a SOURCE-PROGRAM - made with the BASIC-EDITOR - without errors. And so the complete HELP 128 PLUS C 64 commands can be used for making an ASSEMBLER SOURCE PROGRAM, mainly: GENLINE, RENUMBER, APPEND, DELETE, FIND and PAGE-LIST.

2 PASS ASSEMBLER means: An ASSEMBLER-SOURCE-PROGRAM is translated in 65xx machine language code. The SOURCE PROGRAM which is on a diskette, is read for two times. At the first pass the address of all LABELS which are in program are found out. At the second pass all is translated in machine language code.

Before calling the ASSEMBLER, an ASSEMBLER-SOURCE-PROGRAM with the BASIC-EDITOR must be made. This program contains instructions line by line which are translated by the ASSEMBLER. Each line has a line number like a BASIC-COMMAND LINE which can be from 0 - 63999, similar to a BASIC

program. An ASSEMBLER LINE contais in opposition to BASIC only one command. An ASSEMBLER LINE must look like this:

LINE NUMBER (LABEL) OPCODE OPERAND (REMARK)

Each line must also have an OPCODE (COMMAND). The bulk of the OPCODE (COMMANDS) still need an OPERAND (ADDRESS) to be able to execute this command. In such a case there must also occur an OPERAND (ADDRESS). A LABEL and a REMARK need not to occur. For each line only 1 LINE NUMBER, 1 LABEL, 1 OPCODE and 1 REMARK is allowed and between the single kinds there must be 1 space at least.

L A B E L :

It has 1 to 6 digits and must begin with a letter. A LABEL may only consist of letters and numbers. However, you are not allowed to use one of the 56 reserved OPCODES (COMMANDS) and the reserved single letters: A, X, Y of the 65xx MICROPROCESSOR. A LABEL in a program is used if you want to mark a program part eg: sub-program. If the ASSEMBLER recognizes a LABLE during translating, the present memory address (PROGRAM POINTER) will be allocatet to this LABLE. If at this OPERAND (ADDRESS) instead of a numeric address this LABLE is used as a SYMBOLIC ADDRESS, then the ASSEMBLER put in the ADDRESS which has been allocated to this LABEL. If a LABEL has not been defined at this moment in the first PASS, however, occurs as SYMBOLIC ADDRESS then the ASSEMBLER reserves 2 BYTES for this ADDRESS. And so all ZERO-PAGE LABELS must be defined as OPERAND before they occur because a ZERO-PAGE ADDRESS needs only 1 BYTE.

O P C O D E :

Mnemonic for the 65xx microprocessor. It consists of 3 letters and there are 56 commands. Except these 56 commands the HELP 128 PLUS C 64 ASSEMBLER still knows the ASSEMBLER-DIRECTIVES (PSEUDO OPCODES).

The 65xx-microprocessor knows the following 56 OPCODES:

ADC = adding with carry
AND = AND-connection
ASL = moving 1 BIT to the left side BIT 0 = 0
BCC = relative jump if C in status = 0
BCS = relative jump if C in status = 1
BEQ = relative jump if Z in status = 0
BIT = checking for BIT 7 and 6
BMI = relative jump if N in status = 1
BNE = relative jump if Z in status = 1
BPL = relative jump if N in status = 0
BRK = program interrupt I and B in status will be 1 (NO OPERAND)
BVC = relative jump if V in status = 0
BVS = relative jump if V in status = 1
CLC = C in status will be 0 (NO OPERAND)
CLD = D in status will be 0 (NO OPERAND)
CLI = I in status will be 0 (NO OPERAND)
CLV = V in status will be 0 (NO OPERAND)
CMP = compare memory with accumulator
CPX = compare memory with X-register
CPY = compare memory with Y-register
DEC = decrement memory (memory - 1)
DEX = decrement X-register (X - 1) (NO OPERAND)
DEY = decrement Y-register (Y - 1) (NO OPERAND)

EOR = EXCUSIV-OR-connection
INC = increment memory (memory + 1)
INX = increment X-register (X + 1) (NO OPERAND)
INY = increment Y-register (Y + 1) (NO OPERAND)
JMP = JUMP absolut or indirect
JSR = sub-program is called
LDA = load accumulator with memory content
LDX = load X-register with memory content
LDY = load Y-register with memory content
LSR = moving 1 BIT to the right side BIT 7 = 0
NOP = no operation (NO OPERAND)
ORA = OR-connection
PHA = accumulator in the stack (NO OPERAND)
PHP = status in the stack (NO OPERAND)
PLA = accumulator of the stack (NO OPERAND)
PLP = status of the stack (NO OPERAND)
ROL = moving 1 BIT to the left side BIT 0 = CARRY
ROR = moving 1 BIT to the right side BIT 7 = CARRY
RTI = return of interrupt (NO OPERAND)
RTS = return of sub-program (NO OPERAND)
SBC = subtracting with carry
SEC = C in status will be 1 (NO OPERAND)
SED = D in status will be 1 (NO OPERAND)
SEI = I in status will be 1 (NO OPERAND)
STA = load accumulator in memory
STX = load X-register in memory
STY = load Y-register in memory
TAX = transfer accumulator to X-register (NO OPERAND)
TAY = transfer accumulator to Y-register (NO OPERAND)
TSX = transfer stack-register to X-register (NO OPERAND)
TXA = transfer X-register to accumulator (NO OPERAND)
TXS = transfer X-register to stack-register (NO OPERAND)
TYA = transfer Y-register to accumulator (NO OPERAND)

O P E R A N D:

An OPERAND is the address part of an OPCODE. The 65xx microprocessor knows 13 kinds of addressing. By some of the 56 OPCODES you can use more addressing modes, by some only one and there are OPCODES which need no address because the command is occured within the processor. For the 13 addressing modes mentionned above the memory addressing - which is seized at the processor - is 16 BIT or 2 BYTE long. But if you have a ZERO-PAGE address only 1 BYTE must be used for the address because the higher 8 BIT are all 0. For these one or two BYTE-addresses the following figures can be used:

DECIMAL NUMBERS: An address which begins with a number is interpreted as decimal number. The number must be between 0 and 65535.

HEXADECIMAL NUMBER: An address which begins with the Dollar-character $ is interpreted as hexadecimal number. The number must be between $0000 and $FFFF.

OCTAL NUMBER: An address which begins with the @-character is interpreted as octal-number.

BINAER NUMBER: An address which begins with the %-character is interpreted as binaer number. Length of the number up to 16 digits and the number can only be 0 or 1.

ASCII NUMBER: A digit or a character which follows after a ' or a " is transformed to an ASCII CHR$ CODE.

SYMBOLIC ADDRESS: The address which belongs to this symbol is used as numeric address.
The OPERAND can also be joined of more of this numbers. Between the single numbers, however, there must be a + (for adding) or a - (for subtracting).

If only the low-part of a number or symbolic address is needed, so at the end of this number or address there must be the "smaller character" <.

If only the high-part of this number or symbolic address is needed, so at the end of this number or address there must be the "greater character" >.

A D D R E S S I N G M O D E S:

1. IMMEDIATE: The datas are in the BYTE following the command. This addressing mode is 2 BYTE long. First BYTE = OPCODE; second BYTE = DATAS. These datas can be between 0 and 255. In the Assembler-source the OPERAND must begin with the #-character and is only allowed to be 1 BYTE long. eg: LDA#$20 (load accumulator with the NUMBER $20 = decimal 32)

2. ABSOLUT: in the 2 BYTE there is the low-part and in the 3 one there is the high-part of the address where the datas are. In the Assembler-source the OPERAND can be 2 BYTE long. Eg: STA $0374 (Saving of the accumulator in the memory location $0374)

3. ABSOLUT,X: Similar to ABSOLUT, but the content of the X-register is added to the address-value. Then the result is the right address. In the ASSEMBLER-source the OPERAND must be followed by ,X. Eg: CMP $0400,X (Compare accumulator with the memory location $0400+X-register)

4. ABSOLUT,Y: Similar to ABSOLUT,X; only instead of X-register the Y-register. In the Assembler-source the OPERAND must be followed by ,Y. Eg: AND $1000,Y (AND-connection accumulator with the memory location $1000+Y-register)

5. ZERO-PAGE: In the 2 BYTE there is the low-part of the address. The high-part is 0. This kind of addressing is 2 BYTE long and addresses the first 255 memory locations. If the OPERAND assumes a value in the range 0 to 255 and the OPCODE knows this addressing mode, at assembling automatically this addressing is assumed. In the Assembler-source the OPERAND is equal to the absolut addressing.

6. ZERO-PAGE,X: Similar to ZERO-PAGE, only the content of the X-register is added to the address value. If the value is greater than 255, 256 is subtracted. Moreover similar to ABSOLUT,X.
7. ZERO-PAGE,Y: Similar to ZERO-PAGE,X; only instead of X-register the Y-register.

8. INDEXED INDIRECT: This addressing mode is 2 BYTE long. The 2 Byte is a ZERO-PAGE address. To this address the X-register is added. The content of the calculated memory location is the low-part of the address, where the datas are. The next memory location is the high-part of the address, where the datas are. In the Assembler-source there must be at the beginning a paranthesis opened (and after the address a,X). Eg:LDA ($20,X).

9. INDIRECT INDEXED: This addressing mode is 2 BYTE long. The second Byte is a ZERO-PAGE address. The contents of this address shows the low-part, the contents of the next memory location shows the high-part of the address. To this ascertained address the Y-register is added. The result is the address where the datas are. In the Assembler-source the OPERAND must begin with a parenthesis opened (and end after the address with) ,Y.
Eg: STA (160),Y (160 - $A0)

10. RELATIVE: It is at RELATIVE-JUMPS the 2 BYTE. Is the condition complied, so the contents of the 2 Byte is added to the program pointer. Is the contents greater than 127 so 256 are subtracted from the result. In the Assembler-source you only have to attend that this area is kept. Eg: BCC 20 (If C in status = 0 then jump +20)

11. INDIRECT: Is only at OPCODE: JMP possible and is 3 BYTE long. The contents of the 2 BYTE is the low-part, the contents of the 3 BYTE is the high-part of a pointer-address. The contents of this address is the low-part, the contents of the next memory location is the high-part of the address which is jumped on. From this address the program is continued. In the Assembler-source the OPERAND must begin with a parenthesis opened (and end with a parenthesis closed). Eg: JMP ($0302)

12. ACCUMULATOR: Is only possible at moving-commands and is only 1 BYTE long. In the Assembler-source at the OPERAND there have only to be an A.
EG: ASL A

13. IMPLIED: At this addressing mode NO OPERAND is allowed because only within the processor, register are modified. EG: TXA

A S S E M B L E R - D I R E C T I V E S:

Additional to the 56 OPCODE the HELP 128 PLUS C 64 ASSEMBLER knows further 7 ASSEMBLER-DIRECTIVES. There are: the sign of equality = and 6 PSEUDO-OPCODES. Each of these PSEUDO OPCODES begins with a POINT.

BYTE: Here a number between 0 and 255 can be entered. The kind how this figure can look like in the Assembler-source is like syntax of the OPERAND (DECIMAL, HEXADECIMAL, OCTAL, BINAER, ASCII, SYMBOLIC and it can also be connected with + or -). After the instruction .BYTE there also can be more BYTEs, but these must be seperated with a comma. Eg:
.BYTE $20,'A,220,%1101,TXTABL

.WORD: Syntax like .BYTE, however, a number from 0 to 65535. This number is translated in this form that first the low BYTE is written in the memory location and then the high BYTE. Eg: .WORD $FFD2 is saved as D2 FF.

.TEXT: After this command there must be either the '-character or the "-character. With the character with which it has begun, it must also end. Between them can be more characters (letters, figures, special characters). Each of these characters is transformed in an ASCII figure.
Eg:
.TEXT 'HELP 128 PLUS C 64' or .TEXT "2-PASS ASSEMBLER"

.DISP: Like the command .TEXT. However, the characters are transformed in a 6 BIT ASCII. This means that the ASCII code is for 'A $41 and the 6 BIT ASCII 6 code is for 'A $01.

.END: If this command is in the source-program then the ASSEMBLER terminates translation. This command must not be written because the ASSEMBLER writes this itself, if it get a program-end-status at reading in.

.CORR: This command has the following meaning: The machine language code - translated by the Assembler - is immediately written to the considered memory location (POKE). Now, if this memory location is not free, eg: through a ROM or is covered from the operation-system, that is the area from $0000 to $07ff and from $D000 to $Dfff, and also through a LABEL-TABLE, which needs for each LABEL 8 BYTES from $0000 upwards, so here the area where to write the machine language program can be modified through the input of a number up to 4 digits. The entered number is DEDUCTED from the real address (SUBTRACTED)!. Eg: A program should begin at the address $A000 (40960 decimal). Because here is the HELP C-64, through the input of 1000 (=$1000)at HEXA KORR-POKE now the translated program can be written $1000 lower, that means: beginning at $9000).

.LOAD: With this command a new program is called for translation - with the program name following this command. RESTRICTION: The program name must exactly be so long as the program name of the first program!

SIGN OF EQUALITY = : An important sign because you can appoint SYMBOLS to a memory location or you can put the program pointer. There are the following possibilities with the sign of equality:

1. Before the = there is a LABEL: After the = there must be an address number in that form as at OPERAND (DECIMAL, HEXADECIMAL, OCTAL, BINAER, ASCII, SYMBOLIC). The value after the sign of equality appointed to the LABEL before the sign of equality. Eg: LABEL = $0400

2. Before the = there is the special character * (* means: PROGRAM POINTER or current PROGRAM MEMORY LOCATION): After the = there must be an address number in that form as at OPERAND. The PROGRAM POINTER is put to the value indicated after the sign of equality. This command line MUST always be at the beginning of a translation otherwise the memory location, where the program should begin, is undefined. Eg: * = $7000 : The machine language program is saved from $7000 upwards. The programpointer is make up to $7000.

R E M A R K:

A REMARK must begin with a semicolon ;. If the Assembler recognizes this charact r so translation is terminated in this line. All further characters can be used for explanation of this line.

ASSEMBLING

After typing the command key: [the Assembler reports with the question for the program name. The name entered here must be the same as the name of the program existing on the diskette. Further, this name is used as "page-header" at a print-out by the printer.

The second question is for the print-out code. There exist 3 input locations.

1. LOCATION: print-out ERROR with line number. If a blank or 0 is put in no ERROR print-out follows both on screen both on printer. Nevertheless the founded ERRORS are displayed at listend. At input of 1 each ERROR is

indicated.

2. LOCATION: print-out ASSEMBLER LISTING. If a blank or 0 is put in, then no ASSEMBLER LISTING is displayed both on screen both on printer. This can be used, if only ERRORS shall be listed to have a better possibility for corrections. At input of 1 each line is listed both on screen both on printer.

3. LOCATION: print-out SYMBOL TABLE (LABEL). If a blank or 0 is put in, then no SYMBOL TABLE is printed out. At input of 1 at the end of translation an alphabetic sorted SYMBOL TABLE with all SYMBOLS and the addresses belonging to them is listed both on screen both on printer.

For each display on screen there also follows an output on printer (DEVICE 4). At the printer runs a line counter with 72 lines for each page and a page counter. At each new page there follows a head-line output with program name and page. Should no output follows on the printer, so you need only to turn off the printer.

After input of these 3 questions, the ASSEMBLER begins to assemble (translat). During the first pass no print-out follows on screen or on printer. Then at the second pass the program is listed and written to the memory location.

At the end of the translation there is displayed: number of the translated lines, number of the founded SYMBOLS (LABEL) and number of the found errors.

E R R O R M E S S A G E S:

The HELP 128 PLUS C 64 ASSEMBLER can distinguish at translating 6 kinds of errors. With the exception of the error: DUPLICATE SYMBOL which is indicated in the first pass, in the second pass all other errors are indicated - which has been found. If at the question: PRINT-OUT CODE on the first location a 1 has been entered, then each recognized error is printed out with errortext and line number. An error counter counts each error and lists the result at the end of the translation (The error counter counts up to 255 errors and if there are more than 255 errors the counter begins with 0 again).

The ERRORS:

SYNTAX ERROR: Is indicated if no or several OPCODES are in one line. Or if no space is between the entries.

ONE BYTE RANGE ERROR: Is indicated if a figure is higher than 255 but only 1 BYTE is possible for this number. Eg: At ZERO-PAGE addressing.

RELATIVE BRANCH ERROR: Is indicated if at an RELATIVE-BRANCH ADDRESS the branch address is not in the range from +127 and -128 of the program pointer.

ILLEGAL OPERAND ERROR: Is indicated if the OPERAND (addressing mode) is unknown or is not possible for the OPCODE existing in this line.

UNDEFINED DIRECTIVE ERROR: Is indicated if you begin with a point, but the following text does not correspond with one of the PSEUDO-OPCODE-TEXTS.

UNDEFINED SYMBOL ERROR: Is indicated if at a SYMBOLIC ADDRESS occur a

SYMBOL which has not been defined as LABEL.

DUPLICATE SYMBOL ERROR: Is indicated if a SYMBOL has been appointed to more LABELS.

SAVING of MACHINE LANGUAGE PROGRAMS with C 64:

Has the 2-PASS ASSEMBLER translated a program without errors, so you find the translated program in the work-memory of the computer. Now the machine language program can be started with a SYS-command. But mostly it shall be saved for later uses. Now this can happen so, that the POINTER for BASIC-PROGRAM START ($2B = low-part, $2C = high-part) are modified in that way that they get the address of the start of the machine language program. The POINTER for BASIC-PROGRAM END + 1 ($2D = low-part, $2E = high-part) must get the address of the END of the machine language program + 1. Then this program can be saved like a BASIC program. After termination of saving, you are not allowed to forget that the POINTER get their former values. Then this machine language program can only be loaded with the SECONDARY ADDRESS 1 or with the DOS-LOADING COMMAND %.

SAVING of MACHINE LANGUAGE PROGRAMS with the C 128:

After the correct translation a BSAVE command will be prompted. Confirmation by pressing RETURN will lead to execution.

SUMMARY

COMMANDS for BASIC:		
APPEND	: #A	appending of several programs
BASIC-RENEW	: #B	RENEW of a program
COMPACTER	: #C	compacting of a program
UNCOMPACTER	: #C0	uncompacting of a program
DELETE	: #D	deleting of several programlines
END TRACE/STEP	: #E	terminates SINGLE STEP and TRACE
FIND	: #F	searching for entries
GENLINE	: #G	automatically output of linenumbers
HELP	: #H	listing of the last executed line
KILL	: #K	terminates HELP 128 PLUS C 64
PAGE-LIST	: #L	listing of pages forwards and backwards
MATRIX DUMP	: #M	listing of all matrix-variables(arrays) with contents.
RENUMBER	: #R	renumbering of a program
SINGLE-STEP	: #S	execution in single steps
TRACE	: #T	slow execution of a program
CHECKING for UNDEF'D	: #U	program is checked for UNDEF'D STATEMENT
VARIABLE DUMP	: #V	listing of all variables with contents
CONVERSION HEXA-DECI	: !#	converting of HEXA to DECIMAL
CONVERSION DECI-HEXA	: !$	converting of DECIMAL to HEXA
PRINTFRE(x)	: *	indicates free memory
CMD	: (	opens a FILE with CMD
CMD END	:)	closing of CMD and FILE
MODIFYING of DEVICE	: £	modifying of the DEVICE NUMBER
DISASSEMBLER-MONITOR:	:]	
MODIFYING ADDRESSES	: +	
MODIFYING DATAS	: /	
OUTPUT on the SCREEN	:]	
OUTPUT on the PRINTER	: [	
C-64 MEMORY	: >	
FLOPPY MEMORY	: <	
INCREMENT ADDRESS	: RETURN KEY	
DECREMENT ADDRESS	: ↑	
RUN	: *	
DISASSEMBLER 1 LINE	: -	
DISASSEMBLER CONT.	: SPACE KEY	
TRANSFER	: @	
RETURN to BASIC	: =	
DOS-COMMANDS:		
LOAD PROGRAM w Rel.	: /	
LOAD PROGRAM w/o Rel.	: %	
LOAD PROGRAM + RUN	: ↑	
SAVE PROGRAM	: ←	
CHANNEL 15 COMMANDS	: @ with TEXT	
STATUS INPUT CHANNEL 15	: @ without TEXT	
GET CHANNEL 15	: > without TEXT	
VERIFY	: <	
TABLE OF CONTENTS	: $	
2-PASS ASSEMBLER	: [	
PRINT-OUT CODE 1. L	: ERROR	LISTING 0 = no 1 = yes
2. L	: ASSEMBLER	LISTING 0 = no 1 = yes
3. L	: SYMBOLS	LISTING 0 = no 1 = yes