SM-KIT©

A Programming Productivity Tool For Commodore Computers

COMMODORE 8032
COMPUTER

A 4K ROM with both programming and disk handling aids

The SM-KIT is a collection of machine language firmware programming and test aids for BASIC programmers. SM-KIT is a 4K ROM (twice the normal capacity) which you simply insert in a single ROM socket on any BASIC 4 CBM/PET–either 80 column or 40 column. Includes both programming aids and disk handling commands.

ERROR DETECTION: the SM-KIT automatically indicates the erroneous line and statement for any BASIC program error.
LINE NUMBERING: the SM-KIT automatically numbers BASIC statements until you turn the function off.
SCREEN OUTPUT: the commands FIND, DUMP, TRACE and DIRECTORY display on the CRT while you hold the RETURN key (display pauses when the key is released). Continuous output is selected with shift-lock.
OUTPUT CONTROL to DISK or PRINTER: in addition to displaying on the CRT, you can direct output to either disk or printer.
HARDCOPY: allows screen displays to be either printed or stored on disk.
FIND: searches all or any part of a program for text or command strings or variable names. Either exact search or wild card search supported.
RENUMBER: the SM-KIT can renumber all or any part of a program. The selective renumbering allows you to move blocks of code within your program.
VARIABLE DUMP: displays the contents of floating point, integer, and string variables (both simple and array). Can display all variables or any selected variables.
TRACE: SM-KIT can trace program execution either continuously or step by step starting with any line number. Selected program variables can be displayed while tracing.
DISK COMMANDS: as in DOS Support (Universal Wedge), the "shorthand" versions of disk commands may be used for displaying disk directory, initializing, copying, scratching files, load and run, etc.
LOAD: SM-KIT can load all or part of BASIC or machine language programs. It can append to a program in memory, overwrite any part of a program, load starting with any absolute memory location, and load without changing variable pointers.
MERGE: allows merging all or any part of a program on disk with a program in memory.
SAVE and VERIFY: SM-KIT provides one step program save and verification. It also allows you to save any part of a program, or any address range.

Developed by (and available in Europe from) SM Softwareverbund-Microcomputer GmbH, Munchen, Germany

Distributed by: **A B Computers** Colmar, PA 18915

SM-KIT USER'S GUIDE

Table of contents

0. FORWARD

The SM-KIT is a collection of programming and test aids for BASIC programmers using BASIC 4.

In order that you obtain fullest use of SM-KIT, please read the following manual carefully, following the examples where given.

We have fully tested the SM-KIT and think it is error free. But as always, especially with a 4k machine language program that is an extension of the Operating System, it is still possible that unforseen results may occur. If an error does occur, please document it as fully as possible and send it to us. We continually update our programs and would appreciate the input.

We hope SM-KIT proves very useful to you.

Munich, March 1981

SOFTWAREVERBUND MICROCOMPUTER GMBH
Scherbaumstrasse 29
8000 Munchen 83 GERMANY

0.1 INSTALLATION OF THE SM-KIT

The SM-KIT ROM is physically installed in the last free socket in your CBM/PET. Be sure SM-KIT is installed in the correct direction (with the notch the same as existing ROMS), and protect against static electricity.

On the 80 column logic board, the socket is UD12.

On the 40 column logic board, the socket is UD3.

The hexadecimal address for either version is $9000.

Please note that on some of the newer 40 column boards, particularly those with the 12" CRT, the logic board will be similar to the 80 column one, with the SM-KIT installed in UD12.

0.3 START-UP and SHUT-DOWN

START-UP: **SYS 36864** (or SYS 9*4096)

After entering this instruction the system should reply with:

```
SM-KIT
READY
```

SM-KIT commands begin in column 1 on any screen line. They are generally entered by pressing the return key.

To restore the cursor to column 1, you may use shifted RETURN at any time

SHUT-DOWN: **.X**

0.4 SELECTING LOWER CASE or GRAPHICS

.U changes (toggles) the mode switch between lower case and graphics mode.

0.5 REPEATING KEY CAPABILITY

SM-KIT enables keys to repeat automatically if you do not have one of the CBM/PET versions with full or partial auto repeat.

The original BASIC 4 Graphics PET will have repeat functions with SM-KIT.

In the 16B and 32B machines, SM-KIT will provide you with repeat capability by activating the REPEAT key.

The newer 4016/4032 machines with revised logic boards (generally with 12" CRT) already have repeating cursor/edit functions and will not be changed. The repeat capability for the 8032 is also unchanged.

1. OUTPUT OPTIONS

The SM-KIT allows you to direct screen output or SM-KIT command results to either the printer or diskette. These options are described in the following section.

1.1 Output Control

The commands FIND, DUMP, TRACE, and DIRECTORY (@$) will only display output while you depress the RETURN key. Output stops when you release, and continues when again pressed.

You may display continuously by depressing both RETURN and SHIFT-LOCK (remember to release the SHIFT-LOCK when finished).

The STOP key will interrupt the FIND, DUMP, and DIRECTORY functions and put you back in READY mode. With the TRACE command, when you release the RETURN key, you automatically revert to input mode.

1.2 Output to the Commodore Printer 2022/4022

.← (left pointer)

This command sets the flag for the standard printer 2022 or 4022. Every command which produces output will display the output on the screen, then print the screen image on the printer (the one exception being the DIRECTORY command).

As this output is especially directed to the 2022 printer, a character code 17 is sent with every print line so that the printer will function in lower case when the computer is set in upper/lower case mode.

For printers other than the Commodore unit, use the other peripherals command, as this control code may cause unexpected reactions in other printers.

.# (number sign) discontinues sending the output to the printer.

1.3 Output to Other Peripherals

.← (left arrow) la

points the output to the logical address la. You must also use the OPEN command before the first output.

It is only necessary to use la after the left arrow the first time. After that the SM-KIT will assume the same logical address.

.# (number sign)

discontinues sending output to the printer.

1.4 HARDCOPY

This command allows you to print or save chosen display lines.

The format of the HARDCOPY command is:

.* (dot asterisk) (return)

to indicate the first output line and then again for the last line of output. The only catch is that it is inclusive of the first line with the .*, but exclusive of the last line. To output on the standard 2022 printer, nothing else is required other than the two (.*) to indicate the beginning and end of what you have selected. For other peripherals, you must have first used the previous command of .--(left pointer)la giving the logical address for the output.

To output the complete screen display, it is only necessary to enter the .* (return) twice in the first line.

It is not possible to display screen output through screen line 24, since entering .* on line 25 causes the screen to scroll.

When you output onto the diskette, data is stored without quote marks and with CHR$(13) at the end. When you want to read a file stored by the SM-KIT, you must use the GET command, as the INPUT command is ended with a comma or colon.

2. AUTO

The SM-KIT will automatically provide program line numbers when the line number starts in the first column. Line number increments are the same as between the previously given line number.

To try this function, type NEW, then:

```
10 REM FIRST LINE
```

You will notice, the next number will automatically be 20. Type in:

```
20 REM SECOND LINE
```

and number 30 will be provided, then 40, 50, etc.

The number provided can always be changed. If you want an increment greater than 10, simply type over the number provided. For example, if you type 50 over the number 20 provided, the next line number will be 90.

AUTO is turned off by pressing RETURN before entering your basic statement or by starting the line number in the second column. For succeeding lines, AUTO will be smart enough to place the cursor in the second column without providing a line number. Thus it is quite simple to turn AUTO on or off just by beginning the line number in the first or second column.

3. DELETE

With the DELETE command, you can delete specific line ranges from the program currently in memory.

Example:

```
.D100-500    deletes lines 100 to  500
.D-500       deletes all lines until 500
.D100-       deletes all lines from 100 on
```

4. NUMBER (RENUMBER)

The command .N can renumber all or part of a program.

The command can also be used to restructure or move program blocks.

The cross reference list of old-new program numbers can be directed to your printer so that program documentation remains current.

4.1 Format

.N line range,beginning line number,increment

line range as in list format--when no line range is given, the whole program will be renumbered.

beginning line number is the new starting line number and must be given.

increment is a number between 1 and 255 and must be given.

Examples:

.N 100-500,100,10	lines from 100 to 500 in increments of 10 renumbered.
.N 100-,100,5	lines from 100 on in increments of 5 renumbered.
.N -500,1,1	lines until 500 in increments of 1 renumbered.
.N,10,10	entire program renumbered starting with line number 10 in increments of 10
.N 100-500,1000,5	renumbers lines 100-500 starting with new line number 1000, in increments of 5

4.2 Error Messages

? is displayed when the line is empty.

OVERFLOW ERROR is displayed when a line already occurs within the line range given in the NUMBER command. NUMBER will not overwrite or merge this line in, as it will probably cause improper program execution. When this error occurs no part of your program will have been renumbered.

65535 will be provided as a line number when a GOTO line number is undefined and the error message UNDEF'D STATEMENT ERROR would occur. SYNTAX ERROR is displayed when the program is executed or when later the .N is once more executed and encounters line number 65535.

It is suggested that after the .N command, the .F,65535 command be used to search for any undefined statement numbers.

SYNTAX ERROR is displayed when the .N command encounters a line number greater than 63999. This could occur if the .N command had been executed and had renumbered an undefined statement to 65535.

When this error message due to line number 65535 occurs, the program is destroyed because it has been only partly renumbered. Always save a copy of your program, and use the .F command to search for 65535 line numbers.

This error message should not be confused with SYNTAX ERROR when the .N command is incorrectly entered or not started in column 1.

ILLEGAL QUANTITY ERROR is displayed when through .N the specified line range will be exceeded or when the line number will be greater than 63999. In this instance the .N command will not be executed and no renumbering will have taken place.

OUT OF MEMORY ERROR occurs when there is not enough space in memory for the reference table. The table requires 4 bytes per renumbered line. When this error occurs, the .N command will not be executed and no renumbering will have taken place.

4.3 NOTE:

In the case of a large program, the .N command can take up to a few minutes. It is completed only when the cursor reappears.

The .N command is interruptable with the STOP key and renumbering will not have taken place. Even when the reference table is being printed, the STOP key can be used and the program has also not been renumbered.

If the STOP key is pressed after the reference table has been printed, but before the cursor reappears, it will stop the function but will also destroy the program as it has only been partially renumbered at this point.

For a hardcopy of the reference table, use the .← (left arrow) command to direct the output to the printer.

5. HELP

When a program is run and an error occurs, the SM-KIT will automatically display the incorrect line, placing the cursor on the column where the interpreter stopped. Although the error might not have occurred exactly at the cursor position, you can be certain that it is not to the right of the cursor position. In the case of a program line with more than one basic statement, the error will be in the indicated statement, and not in any prior statement on the same 80 character line.

Therefore, it is quite simple to localize a program error. When the error is not easily corrected, the DUMP or PRINT commands can be utilized to help determine the error.

HELP is shut off during TRACE.

6. FIND

With the .F command, your program can be searched for specific text, command strings, or variables. The complete line is then displayed and optionally printed or stored.

6.1 Format

.F line range, search argument

line range is to be given in LIST format but can be omitted. When omitted, the entire program is searched. The comma must be entered, even if the line range is omitted.

search argument can be any basic instruction or character or text string including commas or quote marks. The instruction should immediately follow the comma, as blanks will also be counted as part of the search argument.

When searching for a text string, you must include an apostrophe after the comma. The text string will be searched for only after quote marks or REM statements.

Examples:

.F10-99,A$=	searches in lines 10-99 for A$=
.F,FOR	searches entire program for FOR
.F-99,'ABCD	the text ABCD searched only until line 99
.F,$	searches for all simple string variables
.F,$(	searches for all indexed string variables
.F,%	searches for all simple integer variables
.F,%(	searches for all indexed integer variables
.F,6!!	searches for all 3 digit numbers starting with 6

6.2 NOTES:

Output to the display or printer continues only as long as the RETURN key is pressed. The STOP key interrupts the function while the SHIFT-LOCK key allows the function to continue to the end.

Specified variables will be uniquely found, that is, if .F,A is used, neither AB nor A$ will be identified. Numbers are always unique. The command .F,5 will not find GOTO 50. The wild card character "!" can be used for every position where you do not care what is found. The only restriction is that it cannot be the first character of the search argument and it must be numerical.

When you want the searched for and found program lines directed to the printer or diskette, you must first use the .← (left arrow) command to specify the device.

BASIC instructions are translated to the 1 byte interpreter format before the search begins. Therefore the search argument must also have the format of the abbreviations, as is recognized by LIST. The problem arises mainly with BASIC commands containing the symbols #, $, and (.

The INPUT# and PRINT# commands are stored together as 1 byte. The number sign alone, or INPUT or PRINT alone will not find these instructions. Conversely, the GET# command is stored as 2 bytes, so that GET or the number sign alone will find GET.

Similarily the commands STR$, CHR$, LEFT$, RIGHT$, and MID$ cannot be found with $ alone. The search argument $ will find only simple string variables.

The parenthesis is stored together with the text in the TAB(and SPC(commands. Therefore neither TAB, SPC, nor parenthesis alone will find these two commands.

7. VARIABLE DUMP

.V displays contents of variables used in your program

7.1 Format

.V variable type

.V contents of all simple floating point variables

.V% contents of all simple integer variables

.V$ contents of all simple string variables

.V, array elements of all indexed variables (without contents)

.V,A contents of all elements of the array variable A

.V,A(3) contents of all elements of array from third element on

7.2 NOTES:

For output to printer, first use the .← (left arrow) command.

Output continues only when RETURN or SHIFT-LOCK is pressed. The STOP key interrupts this function.

With indexed variables, output stops when all elements are dumped.

You may use the screen editor to get to the lines displayed by the .V command, then enter .V and RETURN twice to obtain the contents of the variable without again having to type the variable with all its elements.

The DUMP command can display only those variables that have been encountered in the program. The variables are output in the order they occur in the variable list.

8.0 PROGRAM-TRACE

With .R or .G the execution of a program can be observed on the screen and optionally directed to printer or diskette. Additionally, during execution, the contents of several variables can be displayed. The program starting point as well as the trace start can be given. The specified variables can be changed at any time.

8.1 Format

```
.R starting line  Coldstart (RUN)
.G starting line  Warmstart (GOTO)
```

.R is like a start with RUN. It will reset all variables and close all files.

When no starting line number is given after the .R command, the program starts at the first line as with RUN. When a line number is given, (for example .R55), the program starts at this line instead.

When .G without a line number is entered, this has the same effect as CONT and can only be used when the program has already been started with .R or .G with a given line number.

.G with a line number (.G55) means GOTO 55

.G -trace start

The trace start does not have to be the starting program line. Other possibilities for the .G and .R commands are:

.G -500	program continues execution from the current line and begins TRACE from line 500.
.G 20-500	program started with GOTO 20 and TRACE from line 500.

When the RETURN key is released immediately after entering this instruction, the program will automatically stop at the line in which the trace should begin.

TRACE starts only when the given line number is reached. Larger or smaller numbers than the the given line numbers are not taken into consideration, only the specified number is looked for.

When the TRACE start is entered, any subsequent entry of a variable list is ignored.

.G, variable list

In this command, variables can be added after the comma to output desired contents and expressions. The variables and their contents are displayed after execution of each statement.

The variables are separated by commas.

Example:

```
.G,A$,B,C%,D(I),D(I+1),D(3),3*A,CHR$(A)
```

8.2 Array Variables

The variable list is in principle acted upon much like the PRINT command. Therefore, especially with the input of index numbers, two errors from the interpreter can occur.

8.2.1 Indexed variables in the list which are also declared by the DIM statement will cause a REDIM'D ARRAY ERROR.

Example:

```
10 DIM A(20)
20 FOR I=1 TO 20:A(I)=I*2:NEXT
30 REM

.R,A(1)
```

This TRACE instruction results in REDIM'D ARRAY ERROR IN 10. To get around this error, you must enter the following two instructions:

```
.R-20
  20 FOR I=1 TO 20

.G,A(1)
```

8.2.2 The second error occurs when field elements, dependent on the running variable, should be output but the program exits from the loop.

Example:

```
.R-20
.G,A(I)
```

When this instruction is executed for the above program and the loop is fully executed, you will get BAD SUBSCRIPT ERROR IN 30 because A(21) should be output and I after the end of the loop has the value of 21. Before ending the loop, the output of A(I) must be stopped with .G

The same error occurs when, in the same program, A(I+1) or A(I-1) are specified for output. The output of A(I+1) must be stopped before the last execution of the loop. Also, the output of A(I-1) is only permissible after the first execution of the loop; otherwise, ILLEGAL QUANTITY ERROR occurs because of a negative index value.

Output Order
When variables occur in the variable list following the .G command, the contents or results are first output, then the next statement is displayed on the screen and executed. Thus, when you execute a single .G command, the previous contents of the variables are displayed, then the next statement, which is also immediately executed.

Continuous or Step-BY-Step Trace
It is possible to observe a line by line execution of the program by pressing the RETURN key once. The next statement is then listed, executed, and the system responds back with a .G.

As long as you simply press RETURN (on .G line), the last defined variable table is still valid. If you enter .G, the variable table is deleted. With the input of a new variable list, the previously defined variables are also deleted.

With RETURN continually pressed or locked with the SHIFT-LOCK key, TRACE continues to the end or until the program encounters an INPUT instruction.

Trace of GET
The trace of GET is not possible. When a GET will be encountered in the program, you must ensure that the program is able to continue execution, without trace, at that point.

Trace Output on Other Devices
When you want the output to go to another open peripheral, you must use the .G starting line command, as the .R command closes all files.

If you direct output to disk, be careful that the file you open for output is not used by the program you are tracing.

9. DISKETTE COMMANDS

Memory location 1000 contains default peripheral address 8. You may change the default device number by POKEing 1000 with the new number.

9.1 Directory Read

The diskette can be read with the command:

```
.@$drive:data name=type
```

Examples:

```
.@$0          contents of drive 0

.@$1:DAT*     all files whose name begins with DAT

.@$1:*=P      all PRG files

.@$1:DAT*=S   all SEQ file names beginning with DAT
```

The output can be stopped at any time by releasing the RETURN key. The DIRECTORY command can be interrupted with the STOP key.

The output from the DIRECTORY command cannot be directed to the printer, but can be printed later by using the HARDCOPY command. This makes it possible for you to add descriptive text after the data files, making the contents of the diskette easier to understand and find.

9.2 File Handling Format

.@ (at sign) command string

The command string can contain variables, thereby making the commands COPY or SCRATCH easier. Because of this, the Text constants must be surrounded by "quote marks". Normal BASIC string syntax is valid.

Examples:

```
.@"CO:NEWDAT=1:OLDDAT      OLDDAT from drive 1 copied
                           as NEWDAT on drive 0

N$="FILE1

.@"CO:"+N$+"=1:"+N$        FILE1 copied from drive 1 to drive 0 with
                           the same file name
```

Reading Error Messages

```
.@ displays disk status (or error condition codes)
```

10. LOAD

.L is used in the same way as the BASIC instruction LOAD to load a BASIC or machine program. Additionally, the use of a line number after the program name functions as an APPEND command. You may also specify a load address in memory by entering an A with the address after the program name. The variable pointers will not be changed if a semicolon appears after the program name. Therefore a BASIC program and current variables will not be changed by later loading a machine program.

10.1 Format

10.1.1 .L "program name

The .L command has an advantage over the instructions LOAD and DLOAD, in that no parameters must be given. The peripheral 8 is assumed and both drives will automatically be searched for the program. Also after the program name, anything except the semicolon, A, and memory address will be ignored.

This allows you to type .L on the line having the desired program name after you have listed the directory (without having to delete the characters after the name).

Example (after .@$):

```
   16  "SM-KIT8-B7.1F"    PRG
```

Move your cursor to this line, delete one space, type in .L and hit RETURN.

```
.L   "SM-KIT8-B7.1F"    PRG
```

10.1.2 .L "program name", line number (APPEND)

When a line number is entered after the program name, the program will be loaded from that line in memory, serving as an APPEND function. The newly loaded program will be appended to the program already in memory. In order that the program will execute correctly, it is important that line numbers of the newly loaded program be higher than line numbers of the existing program segment.

Example:

In memory: a program with line numbers 10 to 500.
On diskette: a program with line numbers 1000 to 5000.
After the command:

```
.L"DISKPROG",1000
```

the diskette program is loaded as part of the program in memory and can be executed. The command:

```
.L"DISKPROG",300
```

causes the lines 300 to 500 of the program in memory to be overwritten by the diskette program.

It is also possible to leave lines 1000 to 5000 where they are in memory and to

append program lines 100 to 500 with the command:

.L"ORIGPROG",10000

This results in a non-executable program, with the beginning line numbers of 1000 to 5000 and the concluding lines of 100-500! In this situation, you would probably want to use the MERGE command.

NOTE: when you first use the .L function, it is probable that out of habit you will input an 8 at the end of the command, meaning the device number. Unfortunately the SM-KIT understands this to mean you want to load or append from line 8 in memory.

10.1.3 .L "name",A address

Entering a memory address after "name",A loads the contents of a PRG-file after the specified memory location.

10.1.4 LOAD + RUN

The command ↑ (up arrow) allows you to load and immediately execute a program in one instruction:

↑"PROGR" loads and starts program PROGR.

11. MERGE

MERGE will insert and merge together chosen lines of a diskette program with a program in memory. There is no restriction on the line numbers or line order. Each searched for and found line will be inserted exactly where its number belongs. Existing lines will be overwritten.

11.1 Format

```
.M "program name", first line - last line
```

Program name and line numbers (of the program on disk to be merged) are entered as constants. Variables are not allowed.

The line range is given in LIST format. When left out, the entire program is loaded.

12. SAVE + VERIFY

With the .S command you may save and verify a program in one step. The program can be a BASIC program, part of a machine language program, or the contents of a work memory range.

The automatic verify function will respond with either OK (meaning successful save and verify operation), or when it has detected an error, with FILE NOT FOUND ERROR or VERIFY ERROR.

12.1 BASIC Program Save

.S"drive:program name"

Example:

.S"1:program1

saves PROGRAM1 on drive 1 and responds with OK when the save and verify command is successful.

12.2 Line Range Saved

.S"drive:program name", first line - last line

The first and last lines are specified in LIST format.

Saved line ranges can be loaded only with MERGE (.M) or APPEND (.L"name", lines) commands.

Example:

.S"1:PROG",50-200

saves the lines 50-200 as PROG on drive 1.

12.3 Address Range Saved

The .S command can also save chosen memory ranges. This function operates the same as the TIM-Monitor .S, except the address range will be given in decimal form and there is no restriction on name length.

.S"0:MACH1",A32768-34768

saves the screen display as MACH1 on drive 0.

Error Appendix for SM-KIT-17.02.81

1. Printers without Auto-Line-Feed

Printers without auto line feed capability cannot be used with SM-KIT since CR=CHR$(13) is sent, but not LF=CHR$(10).

2. MERGE/DOS SUPPORT

The BASIC 4 version of the DOS-SUPPORT program (UNIVERSAL WEDGE) from Commodore has some errors, that together with the MERGE command produce unwanted results.

When programs are loaded with "/" or "1", DOS SUPPORT sets the program end pointer (42/43) one byte too high. When you store a program that was loaded with the two previous commands, then at the end (after the three zeros) an extra byte is also stored. When the program is next loaded, the added byte is also loaded. When this sequence is performed a few times, there are many unnwanted bytes at the end of the program. When this type of program is loaded in entirety using MERGE, the program can intersect a line in memory, and although the program has been fully loaded, it will not be complete and you will have a mangled program.

When this error occurs, the best way to handle it is to load the program only up to the next to last line with MERGE, and then to key in the last line.

3. AUTO

AUTO increments the next line number by calculating the difference from the last number. When the calculated number will be larger than 65535, the new number will be given as minus 65536, a smaller number than the previous one.

4. Dump of Field Contents

When fields are used and their last dimension is greater than 255, then the continuous function of RETURN or SHIFT-LOCK will not work. This error does not occur with DIM A(1,300) or DIM B(300).

5. TRACE from Floating Point Operations

The step-by-step TRACE will not function with all floating point operations that cause an internal division function. Thus, TRACE will continue until this division instruction, then the following instructions are executed before stopping the trace, even though RETURN was not again pressed.

Example:

```
10 A=3.7
20 A=A/2
30 GOTO 20
```

When this program is started with .R and RETURN pressed only once, TRACE will continue to line 30 before returning with .G.