This is a provisional copy

of the

M A N U A L

for the

CBM 8096

1. System overview of properties of LOS-96 and changes to BASIC-4

Knowledge of BASIC-4 and the handling of CBM 8032 are required to understand these instructions which are merely intended as a supplement to the CBM 8032 handbook.

LOS-96 (Loadable Operating System for 96k Machine) is a BASIC/operating system for CBM 8096 (CBM 8032 with 64k extension board from Commodore). It is compatible with BASIC-4.

LOS-96 is loaded from diskette into the RAM. LOS-96 makes 64k memory available for BASIC (32k for the program, 32k for variables)..

The following points will give a quick overview.

BASIC-stack (unlimited FOR / GOSUB depth)
program change without loss of variables
simplified OVERLAY
ESC-key function for ASCII-CTRL-codes and screen editor
automatic line numbering
deleting groups of lines
BASIC lines up to 250 bytes in length
error message indicating place of error
cassette is no longer supported in the system
default device 8 for LOAD / SAVE / VERIFY
INPUT can read 255 characters
CALL and FUNC for additional loadable machine routines
error in string handling corrected

2. Generel Characteristics

2.1 Auto-Start

If LOS-96 is the first file on a diskette, it can be loaded and started with SHIFT/RUN. Otherwise LOS-96 has to be loaded with BASIC-4 by DLOAD "LOS*" and started with RUN. LOS-96 will then try to load the program "START*" from one of the diskettes from unit 8, and start it. If the program "START*" does not exist, the message FILE NOT FOUND ERROR will appear. This will have no further consequences on the computer or the floppy. The computer will wait in direct modus for instructions.

READY (without period) shows that the computer is not working under BASIC-4 (READY.)

2.2 Loading BASIC-4 Programs

BASIC-4 programs will run with LOS-96, if they are solely written in BASIC. However, since BASIC-4 programs are loaded starting with address 1025 and LOS-96 programs start with 65537, it is imperative to set a comma after LOAD as shown below. This sets the program to the actual beginning of the BASIC program area:

```
     load , "old program"
```

ATTENTION:
When 'old' BASIC programs are loaded without comma, LOS-96 will crash, because the BASIC program will be loaded into the operating system!

```
2.3 Memory Map                                        physic.     logical
128k ---------------------------------------------- 65536       131072
                     vectors and bank
                     select registers
                     ----------------------- 65520       131056
                     ----------------------- (52/53)
                     strings
                     ----------------------- (48/49)
  memory for variables --------------------- (46/47)
                     arrays
                     ----------------------- (44/45)
                     single variables
                     ----------------------- (42/43)
96k---------------------------------------- 32768       98304
96k---------------------------------------- 65536       98304
                     vectors and bank
                     select registers
                     ----------------------- 65520       98288
                     ----------------------- (241/242)
   program memory    BASIC Stack
                     ----------------------- (239/240)
                     ----------------------- (248/249)
                     BASIC program
                     ----------------------- (40/41)
64k --------------------------------------- 32768       65536
64k --------------------------------------- 65536       65536
    ROM area (unused) / I/O ports / screen (like 8032)
32k --------------------------------------- 32768       32768
                      extension for BASIC prg
                      or variable or user prg
                      or system extensions
                      ---------------------- (1252/1253)
                      operating system
   operating system   ----------------------  1024
                      BS-working area
                      -----------------------  768
                      input buffer
                      -----------------------  512
                      processor stack
                      -----------------------  256
                      zero page
0 ------------------------------------------    0
```

2.4 Logical and physical addresses
The following sketch explains the memory allocation:

```
        32-64       64-96      96-132
64k  ----------   --------   ----------
.        ROM     :   PRG    :    VAR
32k  ---------------------------------
       O P E R A T I N G   S Y S T E M
0    ---------------------------------
```

All logical addresses can be reached with the BASIC commands POKE, PEEK, SYS and WAIT. All pointers refer to physical addresses.

2.5 User's Adjustment of Memory Area
The reserved areas for program (32k) and variables (32k) can be extended. A protected area between the top end of the variable area (52/53) and the address 65520, as well as the top end of the program area (241/242) and 65520 can be reserved. (Addresses over 65519 will cause a crash) The bottom ends of the program/variable area can be changed by increasing

the pointers to more than 32768, or decreasing them to less than 32768 when only one of the areas is to be lengthened at the bottom.

2.6 BASIC Stack
The management of GOSUB-RETURN and FOR-NEXT is done by a separate BASIC stack. Therefore there is no limitation of 26 GOSUB levels such as with the CBM 8032. When a program is loaded there should be at least 200 bytes free (fre(0)).

2.7 Program Change Without Loss of Variables/ Overlay
Variables do not have to be deleted when a program memory area is extended. However, when a pointer marks the definition of a function (DEF FN), or READ-DATA, GOSUB-RETURN, FOR-NEXT, and the program is changed later within the region where the pointer is set, the pointers will not change automatically, too, so that they will then read nonsense.
After a change, the program can be started warm (CONT or GOTO) or cold (RUN).

2.8 New Keyfunctions with ESC
Pressing ESC key and another key will have the following effects:

ESC +	
CURS UP	screen scroll down
CURS DOWN	screen scroll up
CURS RIGHT	delete line to the right of the cursur
CURS LEFT	delete line to the left of the cursur
DEL	delete line
INST	insert line
HOME	define the top left corner of the window
CLR	define the bottom right corner of the window
TAB	find the following blank
TAB (SHIFT)	find the following non-blank character
STOP	jump into the monitor (executed at interrupt)
RETURN	move cursur into first column of following line, delete flages, do not take over line

2.9 Automatic Line Numbering
When a line (say line no. 20) is entered by RETURN, then automatically a number will appear with the same increment as before on the following line (say previous line was no 10; automatic next no will be 30), unless this number (30) already appears elsewhere or a number in the interval (20-30) appears elsewhere. In this case no number will be given automatically. An automatic number can be overwritten by the user.

2.10 Input of Very Long Lines
Lines extending over many screen lines should begin in the first or second column; the following screen lines begin in the third column or later.

2.11 Having Many Statements in a Program Line.
More than one statement can be put in one program line. Statements on different lines can be joined in one line, and vice versa.

2.12 Error Messages. The line causing the error will be listed and the cursor will mark the column where the error was detected.

2.13 Cassette Handling and Floppy Pre-Setting. Cassette handling is eliminated completely. The default value for LOAD, SAVE, VERIFY is no longer 1 but 8.

2.14 INPUT can read 250 Bytes.

2.15 ROM. ROM switches the ROM operating system (BASIC-4) on and the RAM operating system (LOS-96) off. RESET is not necessary.

3.1 LOAD
Format: LOAD,;B% address, "drive:name", device #

If ', device #' is not given, the device # is supposed to be 8.
Usually, when a program is loaded into the program memory area it is sufficient to write: load "name"
When load, "name" is typed, the program will be located at the current beginning of the program memory area.
When load ß address, "name" is typed, the programm will be placed starting at the address.
When load ß % address, "name" is typed, the bottom end of the program, which can reach into the variables' memory area will be loaded.
When load; "name" is typed the program-end-pointer will be set after the last byte.
Additional BASIC or machine programs can be loaded by placing a load command into a program.
// NB: The character "ß" stands for "commercial at" //

3.2 SAVE
Format: SAVE ß fromaddress,toaddress,"drive:name", device no
ß fromaddress, toaddress will only cause the part between the two given addresses to be stored. A complete program will be stored by: SAVE "drive no:name", unit no
ß fromaddress, toaddress will only cause the part between the two mentioned addresses to be stored. A complete program will be stored by:
SAVE "drive:name", device no

3.3 VERIFY
The default device no. is 8 unless specified otherwise.

3.4 LIST
Format: LIST from - to (normal listing)
or format: LIST from - to (listing, each statement in a new row)

3.5 DELETE
Format: DELETE 100-200 (abbreviated: deL100-200)
Rows 100-200 (for arguments sake) are deleted.

3.6 FRE
Format: A = FRE (m) m=0: program; m/0: variable
A will contain the number of free bytes.

3.7 CLR
Format: CLR list of variables (variables separated by comma)
The variables (single or arrays) will be cleared from memory. CLR without variables works like CLR in BASIC-4.

3.8 REDIM: changing the length of one-dimensional arrays. Format: like DIM

3.9 RESTORE
Format: RESTORE line, element (, element can be omitted)
The DATA-pointer for READ will be set back to a certain line and a certain element.
line: BASIC line number. element: 1 255

3.10 Machine addresses (POKE, PEEK SYS, WAIT)
The 8096 has got 128k memory, but the processor can only address 64k at a time. POKE, PEEK, SYS, WAIT increase the working memory area.

3.11 Getstring (GET$)
Format: GET$ logical address, string variable, max, end, ignor
Getstring will read from a peripheral device a stringvariable (max. length: 255 characters). Logical address and stringvaraibles must be given. The maximal number of bytes is given by max. The string is disrupted when the end-code end is encountered. The initial code which has to be ignored is given by ignore.
GETSTRING should be preferred to INPUT whenever the delimiters comma, colon and CR shall not apply or when code 0 has to be read.

3.12 Instring (INSTR)
Format: INSTR(chain, stringvariable, column, mode)
A certain chain is searched for in a string variable. The column gives the position where the search is started. The mode gives the direction of the search.

3.13 Midstring (MIDSTR)
Format: MIDSTR(strvar, column) = string
Midstring overrides a string (string) with another string (strvar), starting at the point "column". From there on the first string (strvar) will be overwritten.

3.14 ASC/CHR$
Format: A= ASC(string, column, length)
        A$= CHR$(number, length)
CHR$ will change a number into a 1-5 bytee string, and ASC will change such a string back into a numerical value.

3.15 CATALOGUE$
Format: CATALOGUE$ string, array, device no, drive, sample, mode
CATALOGUE$ transfers the directory of the floppy into an array.

3.16 IF THEN ELSE
Format: IF logical expression THEN yes-possibility ELSE no-possibility
The IF THEN command in BASIC-4 is now extended by ELSE.

3.17 ON ERROR GOTO / EL, EC, EO/ RESUME
Format: ON ERROR GOTO line no
        RESUME   RESUME NEXT   RESUME line no
        RESUME ERROR
ON ERROR GOTO: In case of an error the job will be continued at the given line. The three variables EL (ERROR-LINE), EC (ERROR-CODE), EO (ERROR-OFFSET) contain all information what kind of error occured and where in happened. RESUME indicates how the program is then to be continued (repeat faulty statement, go to the next statment, go to line no., report fault directly and interrupt job).

List of ERROR Messages

EC	EO	ERROR (EO)
-1	0	TOO MANY FILES
-2	14	FILE OPEN
-3	23	FILE NOT OPEN
-4	36	FILE NOT FOUND
-5	50	*SEARCHING
-6	61	FOR
-7	65	*LOAD
-8	70	*VERIFY
-9	77	DEVICE NOT PRESENT
-10	95	*FOUND
-11	102	*OK*
-12	106	*READY*
-13	114	*ARE YOU SURE?
-14	129	*? BAD DISK *
+1	0	NEXT WITHOUT FOR
+2	16	SYNTAX
+3	22	RETURN WITHOUT GOSUB
+4	42	OUT OF DATA
+5	53	ILLEGAL QUANTITY
+6	69	BASIC STACK OVERFLOW
+7	89	OUT OF MEMORY
+8	102	UNDEF'D STATEMENT
+9	119	BAD SUBSCRIPT
+10	132	REDIM'D ARRAY
+11	145	DEVISION BY ZERO

+12	161	ILLEGAL DIRECT
+13	175	TYPE MISMATCH
+14	188	STRING TOO LONG
+15	203	FILE DATA
+16	212	FORMULA TOO COMPLEX
+17	231	CAN'T CONTINUE
+18	245	UNDEF'D FUNCTION

3.18 DISPOSE

Format: DISPOSE NEXT   DISPOSE RETURN
        DISPOSE level   DISPOSE NEXT,RETURN,NEXT, etc.

DISPOSE manipulates the BASIC-stack. DISPOSE NEXT removes the loop, in which the program is, from the stack. DISPOSE RETURN removes the subroutine from the stack. The program does not return to it and the address is forgotten. DISPOSE level (level between 1 and n) sets the stack onto the required level.

3.19 INPUT / PRINT - ß (row, column)

Format: PRINT ß (line, column) list of variables
        INPUT ß (line, column) variable

line: 0 ... 24; column: 0 ...79;
The argument after PRINT and INPUT places the cursor onto the specified position on the screen.

3.20 USING

A$ = USING (format, var-list)
USING is a string function which can insert variables into a string. USING cannot be used in direct mode. The format specification is a string expression.

3.21 INPUT USING

Format: INPUT ß(line, column)
        INPUT USING (format) input-varaible
        INPUT ß(line, column) USING (format) input-variable

input-variable: numerical or string variable, single or array.

4. CALL / FUNC

4.1 OVERVIEW

LOS-96 offers an interface to 'foreign' machine programs, which are called by name via the interpreter. The interface resident in LOS is very simple: when the interpreter receives the token 'CALL' or 'FUNC', he will jump over the vector in 1507/1508.

4.2 The Table

CALL and FUNC call up a machine program via a name. The names and entrance addresses of the programs are given in tables. The vector to the table is in 1510/1511.
The tables are structured as follows: the first part contains the addresses of routines and subtables, the second part contains the corresponding names.

4.3 Calling up CALL or FUNC from the Operating System

When the interpreter notices a CALL statement in the interpreter loop or a FUNC statement in an evaluation of a function, the interpreter will set cell 94 to 1 (for CALL) or 2 (for FUNC) and will jump to the CALL routine via vector 1507/1508

4.4 REM-Routines

The interpreter accepts after REM every byte except 0 (end of line). This property can be useful when short (less than 250 bytes) machine programs have to be writen directly into a BASIC program. In this case no further loading or memory reservation is necessary.