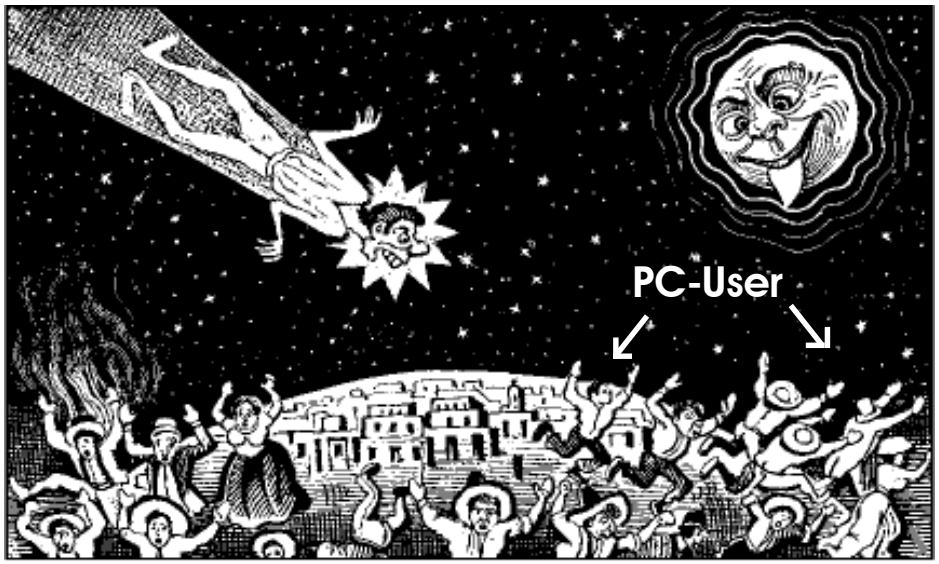
COMET

Z80 Assembler for the SAM Coupe



COMET version 1.3 Manual · 25 July 1991 · Edwin Blink

Booklet by Wolfgang Haller/SPC/Cologne

REMARKS AND COMMENTS

```
        LD   HL,13*256+23    H=depth=13,L=width=23
        CALL drawsquare
        LD   A,22            print AT:
        RST  16
        LD   A,B             line INT (b/8)+1
        RRCA
        RRCA
        RRCA
        AND  31
        INC  A
        RST  16
        LD   A,C             column int (C/4)+1
        RRCA
        RRCA
        AND  63
        INC  A
        RST  16
        LD   DE,message
        LD   BC,messend-message
        CALL &0013           print string 'message'
        POP  BC
        LD   A,C             next column (7 chars)
        ADD  28
        LD   C,A
        CP   254             line full ?
        JR   NZ,repeat       jp if not
        LD   C,2             left column
        LD   A,B             next line (2 chars)
        ADD  16
        LD   B,A
        CP   181             out of lines ?
        JR   C,repeat        jp if not
        LD   HL,&5C3B        FLAGS
wait:   BIT  5,(HL)          key pressed ?
        JR   Z,wait          wait for key
        RES  5,(HL)          reset key
        RET
        INC  "square I.S"    include CALL routines:
                             drawsquare, drawline
message: DEFM "COMET"
messend:
```


INTRODUCTION

COMET is a Z80 assembler designed to make full use of the SAM Coupe's screen and memory capabilities, on both 256K and 512K machines, with at least one disk drive. COMET works with SAMDOS or MASTERDOS.

Features:

- Very fast full screen editor.
- Uses no linenumbers for source.
- Can handle sourcefiles over 400K (512K SAM).
- Objectcode can be put everewere in the 512K internal memory.
- Code files over 400K can be merged from disk in to the objectcode on assembling.
- Source files up to 24K can be included from disk on assembling.
- On line command handler and calculator.

THE EDITOR

The editor works just like a word processor and uses screen mode 3, with 64 characters on a line. There are no line numbers used so you can make full use of each line.

On the screen is a flashing cursor to mark your position. You can only move vertically with the cursor when there is a sourcefile in memory. You are allowed to go one line past the source to extend your source. This method prevents invisible blank lines at the end of the source. Which only take up extra memory.

When the cursor is at the first line of the screen and cursor up is pressed, the screen will scroll down by one line and a previous line is printed. Unless that line is the first source line. This also works for cursor down but then the other way around.

There are a number of keys to make editing easier for you. Look at the following list which shows them all.

CONTROL KEYS

ARROW KEYS	movement of the cursor the arrow points to.
CAPS	toggle caps lock
DELETE	backspace
SYM DELETE	delete character. Characters after the cursor are moved left by one position.
CTRL DELETE	delete a complete line
INV	insert a character
EDIT	clear the current line
SYMBOL EDIT	restore the current line
TAB	tabulate to the right
SYM TAB	tabulate to the left
SYMBOL N	find next item (see find command)
SYMBOL E	insert the instruction EX AF,AF'
SYMBOL I	toggle insert mode
SYMBOL C	enter command/calculator mode
SYMBOL S	Swap location
F7	first source page
F4	page up
F1	page down
F0	last source page
F3	insert blankline
F2	insert blockmarker (see block commands)
F9	")" close bracket
F8	"(" open bracket
F6	"&" Hexadecimal sign
F5	"%" Binairy sign

All other keys work in the same way as in BASIC.

The EDIT key will clear the line and move the cursor to the beginning of the line. If a line is accedently changed or cleared, don't worry if the cursor is still on that line. It's not stored in memory yet. If SYMBOL EDIT is pressed then the line will be restored.
However if the cursor has left that line already, SYMBOL EDIT can not restore the line as the old line has already been changed in memory.

Example:

Enter the following line followed by RETURN.


```
          XOR  A              signal cls entire screen
          CALL &014E          Jp table CLs BlocK
          LD   A,1
          LD   (&5A4D),A      set fatpix
          LD   (&5A55),A      set over 'XOR'

repeat:   LD   BC,0           B=y=0,C=x=0
          LD   HL,&BFFF       H=depth=191,L=width=255

nextsquare:
          CALL drawsquare
          INC  C              x=x+2
          INC  C
          INC  B              y=y+2
          INC  B
          LD   DE,-&0404      depth=depth-4,width=width-4
          ADD  HL,DE
          LD   A,B            Halfway the screen ?
          CP   96
          JR   C,nextsquare   another square if not
          LD   HL,&5C3B       FLAGS
          BIT  5,(HL)         key pressed ?
          JR   Z,repeat       repeat sequence not
          RES  5,(HL)         reset key
          RET
          INC  "square I.S"   include CALL routines:
                              drawsquare, drawline
```

==========

Include demo square2. Includes file "square I.S" (works in MODE 3)

```
          ORG  30000
          DUMP $

          XOR  A              signal cls entire screen
          CALL &014E          Jp table CLs BlocK
          LD   A,-2           channel 'S'
          CALL &0112          open channel
          LD   A,1
          LD   (&5A4D),A      set fatpix
          LD   BC,5*256+2     B=y=5,C=x=6

repeat:   PUSH BC
```

INClude demonstraction

Firstly we create an include file. Enter this source and save it as 'square I'

This include file hold the subroutines 'drawsquare' to draw a square at C,B with width L and depth H and the subroutine 'draw line' to draw a line to B,C

```
drawsquare:
          INC  C
          PUSH BC              plot pixel at B,C
          PUSH HL
          CALL &0139           Jp table PLOT
          POP  HL
          POP  BC
          DEC  C
          LD   A,C             add width to obtain top-
          ADD  L               right corner
          LD   C,A
          CALL drawline
          LD   A,B             add depth to obtain Bottom-
          ADD  H               right corner
          LD   B,A
          CALL drawline
          LD   A,C             sub width to obtain Bottom-
          SUB  L               left corner
          LD   C,A
          CALL drawline
          LD   A,B             sub depth to obtain Top-left
          SUB  H               corner
          LD   B,A

drawline:
          PUSH BC              save coords and depth/width
          PUSH HL
          CALL &013F           JP table DrawTo
          POP  HL
          POP  BC
          RET
```

==========

Include demo square. Includes file "square I.S"

```
          ORG  30000
          DUMP $
```



```
     ;This is a remark
```

move the cursor back on that line with the cursor keys and press EDIT. The line is now cleared. Press SYMBOL EDIT and the line is restored.ie the same as it was before. If you have left the line after you pressed EDIT and go back to the line and then pressed SYMBOL EDIT,the line is not restored.

There are a lot of ways to leave a line and each one of them firstly checks if the line was altered. And the standard procedure of tokenising and space stripping is executed then. With this method it is not required to press RETURN each time a line is changed.

PAGE KEYS

Every page control key. Will move the cursor to the top left corner. Unless the page holds less the 24 source lines. The cursor then is moved to the left side after the last line.
If 'F7' is pressed the 1st 24 lines of the source are printed. on the screen.'F0' displays the last 24 lines of the source.
PAGE UP 'F4' will go back by 24 lines unless the start of the source has been reached.PAGE DOWN 'F1' will advance by 24 lines unless the end of the source has been reached.

INSERT MODE

When insert mode is active a line will be inserted if the RETURN key is pressed. This allowes you to extend or insert instructions easely. The SYMBOL I key-combination toggles this mode.

Example:

Enter the following lines.

```
     ;First line
     ;Second line
```

Go back to the first line. Press RETURN and the cursor will be at the second line. Go back again and press SYM I and then RETURN. A blank line is inserted between the two lines and the cursor is at that line. Each time RETURN is pressed a line is inserted until SYMBOL I is pressed again.

SWAP

The swap function can be usefull if you are working in a large source. It allowes you to switch between different locations in the source.
When SYMBOL S is pressed a marker is set at the 1st line on the screen (not visible). Then a search is made for a swapmarker. If There are no other swapmarkers. The display stays the same. If a swap marker was found then it removed and 24 lines are printed starting at the line the marker wa found.

If SYMBOL S is pressed again the display looks the same as it was when SYMBOL S was pressed for the second last time.

Example:

Enter 48 lines (2 pages) of text. Doesn't matter what. Press 'F7' to go to the 1st page and press SYMBOL S. Nothing seems to happen. But there has. The cursor is printed at the top left corner and a swap marker is put at the first line on the screen (never visible). Now move to the last page by pressing F0 and then SYMBOL S. You're now looking at the last page. Press SYMBOL S again and the first page is displayed.

The swap function will not work if there is no source or when the first line on the screen is at the end of the source. Please note that when swap is used and the first line of the screen you where looking at is altered, then the swap marker is removed. Also when the source is saved, swap markers are removed. The command 'Z' can also be used to remove them.

ENTERING SOURCE

When you enter a source statement correctly. Lower case characters of instructions and hexadecimal numbers are changed into uppercase characters, spaces are removed and the line will be automatically be tabulated to give a neat impression on the screen.

Enter the the following line to see what happens.

```
label:ORG 50000 ;set origin at 50000
```

Will be reprinted like this:

```
label:      ORG  50000 ;set origin at 50000
```


MDAT demonstration.

To demonstrate the MDAT (Merge DATa) instruction, this source will merge a screen into the objectcode. When this code is executed the screen will be displayed untill a key is pressed.
Before you assemble this source insert a disk with a screen on it and enter the file name of a screen in the MDAT statement.

```
          ORG  33000           code must be above 32767
          DUMP $

start:    IN   A,(250)         save LOMEM
          EX   AF,AF'          press SYMBOL E for this one
          LD   HL,screen_start+24576
                               point to palette data
          LD   DE,&55D8        palette table
          LD   BC,40           40 palettes
          LDIR                 move the palette colours
          HALT
          DI
          IN   A,(252)         get screen page
          PUSH AF
          AND  31              keep the page only
          OR   32              RAM at 0 to 16383
          OUT  (250),A         select screen page at LOMEM
          OR   64              MODE 4
          OUT  (252),A         set VPAGE
          LD   HL,screen_start
          LD   DE,0
          LD   BC,24576
          LDIR                 move screen data into screen
          EX   AF,AF'          restore LOMEM
          OUT  (250),A
          EI
          LD   HL,&5C3B        FLAGS

waitkey:  BIT  5,(HL)          has a key been pressed ?
          JR   Z,waitkey       repeat if not
          RES  5,(HL)          reset key
          POP  AF              get VPAGE back
          OUT  (252),A         and restore VPAGE
          RET
screen_start:
          MDAT "screen name"   Enter a screen name here
length:   EQU  $-start
```

Enter your control codes as decimal numbers seperated by commas Up to a maximum of 14 numbers. Default: undefined.

COLOURS

If you wish to use different colours then this option allowes the colours been changed. For each pen colour a palette value from 0 to 127 has to be given. If no flashing colours are required then enter the same number for the second colour as the first number. Press a key and COMET will be saved as an auto file at your disk. Default: col0:0,0 col1:17,85(Flash) col2:34,34 col3:127,127

LABEL JUSTIFY

The labels in the source can be justified to the left side.ie

```
start:        EQU  $
```

Or to the right side of the label field.ie

```
        start:EQU  $
```

Default: left justify

WORKSPACE SIZE

When COMET has been loaded a workspace will be created.Normally a workspace of 5 pages (80K) is created. If you wish to have a different size then enter a new size in pages.

LOADING COMET

Insert the COMET disk and press F9.When COMET has loaded, COMET will be relocated to the first free page at the end of memory the workspace will be put on the pages below that.
COMET will normally be at page 28 on a 512K SAM or page 12 on a 256K SAM. The workspace will normally be in the pages 23 to 27 on a 512K SAM or pages 7 to 11 on a 256K SAM. (workspace 80K).

Finally there are some short demo sources following. Have a look at them. They can be usefull in some way.

The programmer,
Edwin Blink.

The first 15 characters of a line are called the label field. The next 5 characters of the line are called the opcode field. The following 47 characters, the operand field.

L A B E L S

Labels can be up to 14 characters long and are followed with a ":" in the label field. Labels must start with a alpha character. All other characters may be any character except a space, dollar sign '$', percent '%', ampersand '&', brackets '()', minus '-', plus '+', asterix'*', slash '/', back slash '\', comma ',' accent '`', double point ':', semicollon ';' and quote '"' character. Please note that lower case characters are not the same as upper case characters. ie LABEL is not the same as label. Labels may not be equal to a token like 'RET' or 'ret'.

An example of good and bad labels:

```
label_14_chars:EQU  12    ;good.
start:         EQU  12345 ;good.
Command_one:   EQU  1     ;good.
rst:           EQU  207   ;wrong.The label is equal to the
                'RST
label_is_too_long:EQU14   ;wrong.Label is too long.
                           token.
```

When a label is entered incorrectly the line will not be tabulated. Like these:

```
1label:EQU1          ;wrong.The label starts with a number
label$:DEFM"string"  ;wrong.A symbol which is used by the
                      assembler is used.
endEQU$              ;wrong. Missing ":"
```

REMARKS

When you want to enter a remark on a line or after a source statement, you have to start with a semicollon followed with your remark.
The number of spaces before the semicolon are counted and the result is stored after the semicolon (not visible) to keep the same distance between whatever is before the remark and the remark.

C O M M A N D S

Before you can use a command, SYMBOL C has to be pressed first. The line will be cleared and a '>' character is printed at the left side of the line to indicate you are in the command mode. Now you can enter one of the command characters with eventually some parameters. If you wish to leave the command mode then simply press return with no command entered.
If an error occures a message is printed.The first key hit then will bring you back to a blank command line with the key which was hit, printed.If the key was RETURN then the command mode will be left. If a command is executed with no errors the command mode is also left.

Summary of commands:

A	assemble
B	go to basic
C	copy block of source (see block commands)
D	directory
E	erase file
F	find next item (same as SYMBOL N)
F B	find block marker
F E	find error
F word	find word in label field or operand field
F label:	find label in labelfield
G	go to menu
I	view initial control string
I num1,..,num15	define initial control string. Num1 is the number of control bytes. num2 to num15 are the control bytes.
L	Load source
M	Merge source
N	New source
O	Save objectcode
P	print block of source or complete source
P ;	as above but remark markers (semicolons) are printed as a space. Semicolons which do not represent a remark are printed. This option can be used to print simple letters.
Q	Quit. The assembler markers in the page allocation table are removed and the disk will be re-booted
R	relocate block of source
S	save source
T num1,..,num15	send data to printer. Max.15 bytes
U	Undo (delete) block
V	view all labels in 2 columns

SELECT FONT

Press C,A or O

C for current font.
This is the font currently in memory. Normaly the standard SAM font.

A for assembler font.
This font is displayed on the screen above the question.

O for other
This option allowes you to load your own font from disk.

When you choose C or O you will have to wait for a while. The font will be POKEd in an expanded version in the assembler.

ERROR BEEPS

This option gives a beep (actually a soft ping) when you make an error in the editor (when selected of course). Default: on

TABULATIONS

When you want to alter the tabulations of the TAB key. Then Press 'Y' and enter four numbers from 1 to 62. Default: 15,20,26,35

ENABLE PRINTER

If you don't have or don't want to use the printer commands (LIST ON, commands 'P','V*','T' and 'I'), then press 'N'. The printer commands will then be disabled. Default: Printer enabled

LINEFEEDS

When your printer needs linefeeds after cariage returns. Then press 'Y' Default: Linefeed after cariage return

CONTROL CODES

This option allowes you to send a string of control codes to the printer, each time you use the 'P' command. This string can also be changed in the editor. Using the 'I' Command.

B go to Basic ,will do just that

C Change directory. Can only be used if masterdos is booted

D directory,will print a detailed directory of the current drive

E Erase, will give a directory and ask for the file to erase. use D to reprint the directory or press return to return to the menu. If the filename entered ends with a full stop '.' then '.S' is added.

L load ,will give a directory and asks which file you want to load. Press return to use the current name. If the file name is shorter then 9 characters then the extension '.S' is added.

M merge,will merge a file at the end of the source in memory.

S Save,will save the source in memory. Like load and merge. A filename extension '.S' will be added if the filename is less than 9 characters.
If the file already exist then the file on disk will be renamed as a backup version by using the extention '.B'. If there was also a backup version on the disk then that one will be erased.
Swap markers (see swap) will be removed before the source is saved.

N new device,enter the device character followed with the device number. Devices can be Tape, Disk or if MasterDos was booted Network or ramdisk.

O save object code, if the file name is less then 9 characters the extention '.O' is added. The start and length of the object code needs to be given after the filename is defined.

Q quit, Will remove the assembler and workspace markers from the page allocation table,restore the palette and reboot the disk.

R return, returns to the editor.

INSTALLATION

On the COMET disk is also a file called 'COMET inst'. This program allowes some alternations to be made to suit your wishes. These changes will be put in the machine code of COMET.
When the 'COMET inst' file has loaded you will be asked several questions to answer.

V 1 or 2	view all labels in 1 or 2 colums	
V symbol	view all labels starting with 'symbol'	
V 1 or 2,lab	same as above but in 1 or 2 columns	
V symbol:	view label 'symbol' only	
V *	same as above but to the printer	
V *1 to 5	" " " " " " "	
V *symbol	" " " " " " "	
V *1 to 5,lab	" " " " " " "	
v *symbol:	" " " " " " "	
W	view workspace	
W p1,o1,p2,o2	define workspace	p1 start page
		o1 start offset
		p2 end page
		o2 end offset
X address	execute code at'addres' and return to editor	
X page,address	same as above but HIMEM set to 'page'. address' may be a number from 0 to 65535	
Z	Remove swap markers from sourcefile. Note. Swap markers are also removed before the sourcefile is saved.	

Numbers may be decimal, hexadecimal, binair or one character strings. Like: 65, &41, %01000001 or "A"

CALCULATOR

If you wish to make calculations or convert a number to decimal hexadecimal or binair. Enter the command mode and enter the number or calculation and press return. The number or result will be printed like this:

00000 000=Hi 000=Lo &0000 %00000000=Hi %00000000=Lo ASCII' '=Lo

The first number is a 16 bit decimal word of the number/result, the second the high order byte the third the low order byte, the fourth number is hexadecimal notation of the number/result, the fifth the high order biniary notation of the number/result, the sixth the low order binairy notation of the number/result and then the ASCII character. The code for the character is taken from bits 0 to 6 of the number.ie MOD 128. A space is printed for codes <=32.

Numbers can be entered in decimal,hexadecimal or binair.Decimal numbers are entered like normal, hexadecimal numbers must start with a ampersand '&' and binairy numbers with a percent '%'.

Also one character string are allowed.The character must be put between two quotes. If the quote itselve is required then only two quotes must be used. Negative two complements numbers can be entered by putting a minus sign before the number, like -1.

Examples:

```
99      =99             decimal
&1F     =31             hexadecimal
%1100   =12             binair
"A"     =65             character
""      =34             quote
-1      =255 or 65535 negative two complements number
```

You can use additions,substractions,divisions,multiplications and modulus in the following way.

```
8+88    =96             addition
48-16   =32             substraction
12*24   =288            multiplecation
96/16   =6              division
98\9    =8              modulus
```

All these numbers and calculations are also allowed in sourcestatements.

If you want to use the high byte of a label only then do this:

```
LD    H,label/256
```

or if you want to use the low byte only:

```
LD    L,label\256
```

Please note: All calculations have the same priority. 10+10/10 will be two and not eleven.

BLOCK COMMANDS

Before you can use a block command you have to define a block first. To define a block move the cursor to the line which should be the first line in the block and press 'F2'.
Key 'F2' will insert a line with a blockmarker on it. Do not alter this line as the blockmarker will not be reconized anymore as a blockmarker.

Wrong file type	the file is not a code/screen file
Invalid device	Disk only.
Disk error	Error during a disk operation.
File too large	Include file must be smaller than 24K (24576 bytes).
Include in Include	No includes are allowed inside a include file.

WARNING:
Do not press the break button when you are in the editor or during assembling. Pointers will not be updated on re-entry and may be fatal for your sourcefile. However you are allowed to press the break button during execution of objectcode.

CONVERTERS

On the disk are some converters which can be used to convert sources from different assemblers to a COMET source.

SC CONVERTER

If have used the SC_Assembler or have sources of this assembler you can use the program 'SC convert' to make a COMET file of it. Please note that Undocumented instructions are converted but will not be reconized by COMET. Also those assembler commands which start with a asterix will will be converted into a blank line.

LERM CONVERTER

Also on the disk is a converter to convert LERM source files. Multi statements are also converted but each statement will have its on line. The token ENT will be converted into a question mark as this instruction has no use.

BASIC MENU

When you look at the menu there are 5 windows. The upper window holds the PATH of subdirectories if MasterDos is booted else this window is not used. At the right middle side are two small windows the upper one holds the version number. The window beneath that one gives information of the current device, Which filename is used the length of the source file in memory, the length of the symbol table and how many memory is left in the workspace.

The bottom window holds the copyright message. the window at the middle of the screen holds all possible options which you can access with the coresponding keys.

EDITOR ERRORS

Out of memory	there is not enough memory to insert a line.
Not understood	Parameters are incorrect.
Invalid block	Block is empty, undefined or too large.
Inside block	The editor can not copy or relocate a block inside the block.
Number out of range	Number bigger than 255 or 65535.
String too long	string has more than 14 characters or more then 15 numbers are used.
Not found	Item not found.
Invalid Workspace	Start bigger than end or memory pages are allready used.

ASSEMBLER ERRORS

The following errors are printed on the screen if assembling has aborted.

Assembly aborted	Indication that something has gone wrong.
Out of symbol space	Not enough memory to define a label. Assembling was immediately aborted
Check source for errors	Errors have been found during assembling. Assembling was aborted after the first pass or immediately if more then 10 errors are found or error was found on the second pass
Escape pressed	ESC key was pressed. Assembling was immediately aborted
Out of memory	There is not enough memory left. Assembling was immediately aborted.

SOURCE ERRORS

If a error is found in the source during assembling, the error is inserted in the source. If an error occured during INCluding the errors occured are put above the INClude statement.

Bad source statement	A source statement was entered incorrectly.
Bad expression	Numberic expresion is incorrect or a label is too long.
Number out of range	Number must be smaller than 256 or smaller then 65536.
Label not found	Label is not defined.
Multiple label	Label is used more then once.
Displacement out of range	relative displacement of DJNZ,JR or Index is too large.
File not found	file not found.

Then move the cursor bolow the line which should be the last line of the block and press 'F2' again. All lines between the two blockmarkers are then the block.

** Block **	this is a block marker
Block	line was altered.Will not be reconized

Enter the following lines

```
;line 1
;line 2
;line 3
```

Move the cursor to the first line and press F2. The blockmarker is inserted and line one stands below the block marker. Move the cursor below line three and press F2 again.The display looks like this:

```
 ** Block **
;line 1
;line 2
;line 3
 ** Block **
```

The lines one to three are the block. Enter after the second block marker the following line followed by RETURN.

```
;line 0
```

Press SYMBOL C to enter command mode. Press R for Relocate block followed by RETURN. The lines which where replaced and following lines(in this case none) are displayed at the top of the screen. If you press CURSOR UP the display looks like this:

```
;line 0
;line 1
;line 2
;line 3
```

COPY BLOCK to the current line

To copy a block, move the cursor to the line where the block has to be copied to and enter command 'C'. The block is then duplicated and copied to the line the cursor was at. After that the first and folowing lines of the block are printed from

the top of the screen. During this command You might see the screen changing a bit. This is normal as the screen is used as a temporary buffer. This also counts for the move command.

RELOCATE BLOCK to the current line

The relocate command 'R', works almost like the copy command. Instead of duplicating the block, the old block is deleted and moved to the lines where the relocate command was entered. After the command has completed the first 24 lines of the new position of the block in the soure are printed.

DELETE (UNDO) BLOCK

To delete a block, just enter the command 'U' and the block is deleted from the source. The first line and following lines after the deleted block are printed on the screen.
On delete,copy or replace block command an invalid block error can be given if the block is to big (>16384). The block then needs to be cut into smaller pieces.

PRINTING SOURCE

To print a part of the source, make a block first and use the 'P' command. If you want to print the complete source you don't need to use block markers. You can can use the print command at once.
During printing each printed line will be printed at the line the command was given. If you press a key you will be asked to abort printing.

Note. All commands which uses blockmarkers. Will remove the blockmarkers afterwards.

INITIAL CONTROL STRING

This string of control codes is send to the printer when the print source command is used. These control codes can be viewed by using the 'I' command. This string can be defined in the installation program and can be changed in the editor by using the 'I' command followed with numbers.
The first number represents the number of control bytes. Use zero if no bytes should be send. If you want control bytes send then use a number from 1 to 14 followed by the control bytes.

Example:

```
                ORG  30000         ;make code to run at
30000
                DUMP $             ;put code at 30000

start:          LD   A,-2          ;channel -2 (=254)
                CALL &0112         ;open the channel
                LD   DE,mes1       ;DE = start of message
                LD   BC,mes2-mes1  ;BC = length
                JP   &0013         ;print message and exit

mes1:           DEFM "Just an example"
mes2:           EQU  $

length:         EQU  $-start       ;get lenght
```

To assemble enter command 'A' in command mode (SYMBOL C). During assembling the screen is deactivated so the screen will go black and the Z80 runs on the full 6 MHz.
When assembling has been completed. You will look at the same screen as when you first entered COMET. Only in the middle of the screen is some extra information.
When assembling has completed succesfully the message 'assembly completed' is printed.The message 'object code xxxxx Bytes' tells how many bytes are assembled. if more than 65535 bytes where assembled then this number the MODulus 65536 of the real number
If you didn't used multiple DUMP's which can cause gaps then this is also the length of the objectcode.
Finally there is the message 'XXXXX Labels used' which tells you how many labels are defined during assembling.

If the message 'Assembling aborted' apears then there has been an error or more.

When there are errors in the source you can use SYMBOL N to find them. When you corrected an error don't forget to remove the error line. Because when you don't you might wonder whats wrong with the line later on. An error looks like this:

```
 ** Error **  Bad source statement
```

There are no limits to store objectcode or symbols as long as there is memory free to store it.

INC "file name" Include source file "file name". This directive allowes a source file which is on the disk to be assembled at the position the directive was put in the source file which is in memory.
No INCludes are allowed inside a INClude file.also the include file must be less than 24K. The include files are loaded into the screen memory so if you might see some strange things on the screen don't worry this is normal.

Note. both MDAT and INC work only with disk or RAMdisk (MasterDos).

Thera are certain instructions which may be entered in other forms.

The instructions ADD A,.., ADC A,.. and SBC A,.. may also be entered as ADD .., ADC .. or SBC ....=B,C,D,E,H,L,(HL),A,etc.
The directives DEFB, DEFM, DEFS and DEFW may also be entered as DB, DM, DS and DW they are changed into the above form.
Undocumented instructions are not implemented (except SLL). If you want to use them then they should be entered like this:

DEFB 221 for IX or 251 for IY
HL version of the instruction.

Example:

The instuction 'LD IXL,123' becomes:

```
DEFB 221
LD   L,123
```

If you want to give a label the current instruction addres, without a instruction following, you can do it like this:

```
label:   EQU  $
```

or without the EQU $:

```
label:
```

NUMBERS

You can include numbers and calculations in the same way as the calculator can handle. You can also use labels and the dollar sign to use the current instruction address.

FIND COMMAND

To find your way to a surtain part of your source. You can use the find command 'F' which can be used in different ways. The find command on its on will search a earlier defined item from the current line. If the item is found it is printed at the first line following the lines after that line. If it was the last item the 'Not found' message is printed.
You can also use SYMBOL N to find the next item,which is easier to use. To define what you want to find, just type it after the command You can use a 'B' to find blockmarkers, 'E' to find errors (which are inserted if there were errors during assembling), strings or numbers which are longer then one character or a label in the labelfield, if a ":" is added to the label.
When RETURN is pressed the editor will search for the item from the start of the source. If the item is found it is printed at the first line following the lines after that line.Use SYMBOL N to find the next item.

VIEW SYMBOLS

Simply enter V and all labels are printed on the screen in two columns. With their values in decimal and hexadecimal numbers. You can also follow a number or a label or both after the command. The number indicates the number of columns the labels are printed in and the labels starting with 'label' are printed. If you add a ':' after the label, then that label will be printed only.
Put a '*' before the parameters if you want to use the printer If you select condensed elite on your printer you can print in five columns. However if the screen is used as output. Don't use other columns then one or two.
When the ESCape key is pressed, the command is aborted.

W O R K S P A C E

The source and symbol table are in a flexible workspace which can be from as small as 3 bytes and as big as 400 K.
The pages which are used as workspace are marked in the page allocation table at &5100 (&AF for a source page and &AC for the assembler page). So that it is not corrupted when you use some basic commands like COPY or if you use a RAMDISK with MDOS. Enter command W only to view the start and end of the workspace. The line will look like this: W 007,00000,011,16383.
The first number is the start page and the second is the offset in that page. The third number is the end page and the fourth the end offset in that page.
If you press RETURN the workspace is set. If the values are the same as they

were. The source stays in memory..However if the start of the new workspace is different from the start of the current workspace or if the end of the new workspace is smaller then the end of the sourcefile. The sourcefile is deleted !!! The symbol table is always cleared when the workspace is set.

NEW SOURCE

When this command is executed you will be asked if you really want to delete the complete source. You have to press 'Y' to do so or 'N' to abort.

EXECUTE CODE

The command 'X' allowes you to run your assembled code. If a single number is given then the addresses 0 to 16383 hold the ROM0, 16384 to 32767 hold page zero (system page), page 1 at 32768 and page 2 at 49152. Just like in basic.
If two numbers are given then the first number has to be the page to select at 32768(HIMEM)and the second number the address. The lower 32K are the same as before. But 32768 to 49151 holds page 'page' and 49152 to 65535, holds page 'page+1'.
During execution of objectcode, the break button may be pressed. When your code is executed it will return to the editor. If you wish to return to basic or the menu afterwards, then insert this as the end of your source:

```
exit:    POP  BC   ;drop switch routine address
         POP  BC   ;drop assembler page
         POP  BC   ;drop re-entry address
         POP  BC   ;drop switch routine
         POP  BC   ;drop switch routine
         LD   BC,1 ;1=signal goto basic 6=signal menu
         RET
```

THE ASSEMBLER

The assembler works just like any other Z80 assembler. It converts source into objectcode. But some things are different, which I will explain.

ASSEMBLER DIRECTIVES:

You can use some assembler directives. Also known as pseudo opcodes.

ORG address	set origin at 'address'. The source that is going to be assembled will be made to run at this address. If you don't use this directive. The assembler will use 32768 as default.
DUMP address	The code of the source will be assembled at this addres.' Adress' has to be a value from 16384 to 65535.
DUMP page,offset	same as above but allows code been put over 65535.'Page' has to be a value from 0 to 31. Be careful with the page choice to prevent corruption of the Basic system,DOS and assembler. Offset is the address within a page and has to be a value from 0 to 16383.
DEFB num,..,num	define bytes (0-255). This directive allowes you to include data. The data is followed after the directive.
DEFM "string"	define message. This directive allowes you to include text in ASCII form in the objectcode. Note. Don't use quotes inside quotes !!!
DEFS number	define storage. This directive allowes you to create a gap of 'number' bytes. 'number' has to be a value from 1 to 16383 for buffers tables etc.
DEFW numb,..numb	define word (0-65535) like DEFB but now for numbers from 0 - 65535
EQU number	equal or equate. This directive can only be used after a label. It allowes you to give a value to a to a label.
LIST ON/OFF	This directive can be used to print a part or the complete source to the printer during assembling. The line printed will start with the current instruction address followed with the number of instruction bytes and the sourceline. The numbers are printed in hex. Directives do not have instruction bytes. When assembling has completed. Remove this directive from the source. This might save troubles if you assemble again later on.
MDAT "file name"	This directive will Merge a code file (or screen$ file) from disk into the objectcode. There is no limit of the length of this file.