COMET version 1.3 Manual. 12th September 1991

Edwin Blink
Crouwelstr.60
9731 ME
GRONINGEN
HOLLAND.

* * C O M E T * *

Z80 Assembler for the SAM Coupé

I N T R O D U C T I O N

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 either SAMDOS or MASTERDOS.

Program, documentation & demos :	Edwin Blink
Manual editing :	Colin Jordan
Enhancement recommendations :	Colin Jordan Chris White Bruce Gordon

Features:

- Very fast full screen editor.
- Uses no line numbers for source files.
- Can handle source files over 400K (on a 512K SAM).
- Object code can be put anywhere in the 512K internal memory.
- Code files over 400K can be merged from disk into the object code on assembly.
- Source files up to 24K long can be included from disk on assembly.
- On line command handler and calculator.
- Can be used as a simple word-processor/text editor.

Contents

* * S E T T I N G   U P   C O M E T * *

COMET is supplied as a master disk without a disk operating system. To make a working copy of COMET, you need to first FORMAT a blank disk and save onto it as the first file either SAMDOS 2.0 or MasterDOS.

Next, you should also copy onto this formatted disk, the program 'auto COMET' from the COMET master disk.

Now insert your COMET master disk and type LOAD "COMET INST". This will run an installation program which will allow you to change certain features within COMET to suit your taste and system requirements. See page 17 for more details on this.

After answering several questions, you will be prompted to insert your FORMATted disk, and a working copy of COMET will be saved onto it. Please use this disk rather than your original master disk which came with this manual, in case of accidental damage.

You have now created a working copy of COMET. you can now run it by resetting your SAM, inserting your working copy disk, and typing 'BOOT' (or pressing the F9 function key as usual).

COMET represents many months of hard work by several people. Please do not infringe copyright by making illegal copies for your friends. Instead, prompt them to buy their own copy. Software piracy is ILLEGAL, and only results in fewer software companies being willing to develop new software products for the SAM Coupe.

* * T H E   E D I T O R * *

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 editor line.

On the screen is a flashing cursor which marks your position. You may only move vertically with the cursor when there is a source file in memory. You are allowed to go one line past the source to extend it. 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. The following list shows them all.

* * C O N T R O L   K E Y S * *

ARROW KEYS	Movement of the cursor.
CAPS	Toggle caps lock.
DELETE	Backspace.
SYM DELETE	Delete character. Characters after the cursor are moved left by one position.
CTRL DELETE	Delete a 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 blank line.
F2	Insert block marker (see block commands).
F9	")" close bracket.
F8	"(" open bracket.
F6	"&" Hexadecimal sign.
F5	"%" Binary 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 accidentally changed or cleared, there's no need to worry provided the cursor is still on the line - it's not stored in memory yet. If SYMBOL EDIT is pressed, then the line will be restored. However, if you've already moved the cursor from the line, it cannot be restored as it has already been changed in the memory.

example:

Enter the following line followed by RETURN.

```
;This is a remark
```

Move the cursor back to the 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 press SYMBOL EDIT, the line is not restored.

There are a lot of ways to leave a line, and each one of them firstly checks to see if the line has been altered. The standard procedure of tokenising and space stripping is then executed. 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 than 24 source lines, in case the cursor is moved to the left hand side after the last line. If 'F7' is pressed, the first 24 lines of the source file are displayed on the screen. 'F0' displays the last 24 lines of the source file. 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 allows you to extend or insert any instructions easily.
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 useful if you are working on a large source file. It allows you to switch between different locations in the source.
When SYMBOL S is pressed, an invisible marker is set at the first line on the screen. Then a search is made for another swap marker. If one is found, then the next 24 lines after this second marker are displayed on screen and the marker is removed. (If no second swap markers were found, the screen remains unchanged)
If SYMBOL S is pressed again, the display is toggled back to the original.

Example:

Enter 48 lines (2 pages) of text - it doesn't matter what.
Press 'F7' to go to the first page and press SYMBOL S.
Nothing seems to happen - but it has! The cursor is put at the top left corner and an invisible swap marker is put at the first line on the screen.
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 file.
Please note that when swap is used and the first line of the screen you're seeing is altered, then the swap marker is removed. Also when the source file is saved, the swap markers are removed. The command 'Z' can also be used to remove them.

* * E N T E R I N G   S O U R C E * *

When you enter a source statement correctly, lower case characters of instructions and hexadecimal numbers are changed into upper case characters. Spaces are removed and the line will 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
```

It will be reprinted like this:

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

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 are the operand field.

* * L A B E L S * *

Labels can be up to 14 characters long and are followed with a ':' (colon) in the label field.
Labels must start with a alpha character. Any other characters can follow 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 treated the same as upper case characters. ie 'LABEL' is not the same as 'label'
- labels may not be equal to a token or opcode like 'RET' or 'ret'.

Some examples of good and bad labels :

```
label_14_chars:EQU  12     ;Good.
start:         EQU  12345  ;Good.
Command_one:   EQU  1      ;Good.
rst:           EQU  207    ;Wrong. 'RST' is an opcode.
label_is_too_long:EQU14    ;Wrong - label is too long.
```

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

```
1label:EQU1                ;Wrong. Label starts with a number.
label$:DEFM"string"        ;Wrong. A symbol which is used by the
                            assembler ('$') has been 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 semicolon 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 any parameters that are required. If you wish to leave the command mode, then simply press return with no command entered.
If an error occurs, a message is printed. The first key hit will bring you back to a blank command line with the key which was hit displayed. If the key was RETURN, then the command mode will be left. The command mode is left when the command has been executed.

summary of commands:

A	Assemble source.
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 label field.
G	Go to main menu.
I	View initial control code string.
I num1,..,num15	Define initial control code string. Num1 is the number of control bytes. num2 to num15 are the actual control code bytes.
L	Load source file.
M	Merge source file onto end of current file.
N	Clear Source from memory.
O	Save object code.
P	print block of source or complete source.
P ;	as above but ';' are printed as spaces.
Q	Quit the assembler and re-BOOT the disk.
R	Relocate block of source.
S	Save source file.
T num1,..,num15	Send data to printer. Max. 15 bytes.
U	Undo (delete) block.
V	View all labels in 2 columns.
V 1 or 2	View all labels in 1 or 2 columns.
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'address' & return to editor.
X page,address	Same as above, but HIMEM set to 'page'. 'address' may be a number from 0 to 65535.

Numbers may be decimal, hexadecimal, binary or a one character string. Like : 65, &41, %01000001 or "A"

FIND COMMAND

To find your way to a certain part of your source, you can use the find command 'F'. When used without any parameters, it will search for a previously defined item starting from the current source line. If the item is found, it is displayed at the top the screen, along with the source following the point where it was found. If it was the last item, then the 'Not found' message is displayed. You may also use SYMBOL N to find the item in the source. To define what you wish to find, just type what you wish to find after the command. You may use 'B' to find block markers, 'E' to find errors (which are inserted if there were errors during assembly), strings or numbers which are longer then one character, or a label in the label field if a ":" is added to the parameter. When RETURN is pressed, the editor will search from the start of the source. If the item is found, it is displayed at the top of the screen, along with the source following the point where it was found. SYMBOL N finds the next item in the source.

VIEW SYMBOLS

Simply enter V, and all labels are displayed on the screen in two columns with their values in decimal and hexadecimal. You may also follow the command with a number or a label (or both) as parameters. A number indicates the number of columns used for display. If a label is added, the labels starting with 'label' are displayed. If you add a ':' after the label, then only that one label will be displayed. Put a '*' before the parameters if you want to use a printer. If you select condensed elite on your printer, you may print in five columns. However, if the screen is used as output, don't use more than two. When the ESCape key is pressed, this command is aborted.

NEW SOURCE

When this command is used, you will be asked if you really want to clear the complete source from memory. Press 'Y' to do so or 'N' to abort.

EXECUTE CODE

The command 'X' allows you to run your assembled object code.
If a single number is given, then the addresses 0 - 16383 hold the ROMO, 16384 - 32767 RAM page 0 (system page), 32768 - 49151 RAM page 1 and 49152 - 65535 RAM page 2 - as in BASIC.
If two numbers are given, the first number is the RAM page to select at 32768 (HIMEM) and the second number, the address.
The lower 32K is the same as before, But 32768 - 49151 holds RAM page 'page', and 49152 - 65535 holds RAM page 'page+1'.
During execution of object code, the break button may be used.
When your code has executed, you will return to the editor.
If you wish to return to BASIC or the COMET menu afterwards, then insert this at 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 CALCULATOR * *

If you wish to make calculations or convert numbers to decimal, hexadecimal or binary, enter the command mode and enter the number or calculation followed by 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 binary notation of the number/result, the sixth the low order binary 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 displayed for codes <=32.

Numbers can be entered in decimal, hexadecimal or binary. Decimal numbers are entered as normal, hexadecimal numbers must start with an ampersand '&' and binary numbers with a percent sign '%'. Only a single character string is allowed. Characters must be between two quotes. If the quote itself is required, then two empty quotes must be used.
Negative two's complement numbers may be entered by putting a minus sign before the number - eg. -1.

examples:

99	=	99	Decimal.
&1F	=	31	Hexadecimal.
%1100	=	12	Binary.
"A"	=	65	Single character.
""	=	34	Single quote.
-1	=	255 or 65535	Negative two's complement number.

You may use addition, subtraction, division, multiplication and modulus in the following way :

8+88	=	96	Addition.
48-16	=	32	Subtraction.
12*24	=	288	multiplication.
96/16	=	6	Division.
98\9	=	8	Modulus.

All of these numbers and calculations are also allowed in source statements.

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 result in two and not eleven.

Before you can use a block command you need 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'. This will insert a line with a block marker on it. Do not alter this as the block marker will not be reconized anymore as a block marker.
Now move the cursor below the line which should be the last line of the block and press 'F2' again. All lines between the two blockmarkers are now considered to be the block.

```
** Block **    This is a block marker.
**Block**      The line was altered. It will not be recognized!
```

Enter the following lines :

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

Move the cursor to the first line and press F2. The block marker is inserted and line one now stands below the block marker. Move the cursor below line three and press F2 again. The display now 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 were replaced with any 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
```

After any of the following block commands, the block markers are removed from the source.

COPY BLOCK to the current line

To copy a block, move the cursor to the line where the block is to be copied to and enter command 'C'. The block is now duplicated and copied to the line the cursor is at.
The first and following lines of the block are displayed 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 applies to the RELOCATE 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 current line.
After the command is used, the first 24 lines of the new position of the block in the source are displayed.

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 displayed on the screen.

On delete, copy or replace block commands, an invalid block error can be given if the block is to big (>16384). The block then needs to be cut into smaller pieces.

* * P R I N T I N G * *

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 where the command line appeared. If you press a key, you will be asked if you wish to abort printing.

Note. All commands which use blocks will remove the block markers afterwards.

INITIAL CONTROL CODE STRING

This string of control codes is sent 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 number parameters.
The first number represents the number of control code bytes. Use zero if no bytes should be sent. If you want control code bytes sent then use a number from 1 to 14 followed by the actual control code bytes.

USING COMET AS A WORD-PROCESSOR

By using the 'P ;' command, you can print out simple text files which have been encoded as remarks. This command will suppress the first semicolon on the line when the file is being printed, allowing you to use COMET as a simple word-processor.

* * W O R K S P A C E * *

The source and symbol tables are in a flexible workspace which can be from as small as 3 bytes and as big as 400K (on 512K SAMs). 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 BASIC commands like COPY or if you use a RAMDISK with MDOS.
Enter command W on its own 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 unchanged.
However, if the start of the new workspace is different from the start of the previous workspace or if the end of the new workspace is smaller then the end of the sourcefile, THEN THE SOURCE FILE IS DELETED FROM MEMORY !!!

The symbol table is always cleared when the workspace is set.

* * T H E   A S S E M B L E R * *

The assembler works just like any other Z80 assembler. It converts source files into object code. However, a few things are different. These are explained below.

ASSEMBLER DIRECTIVES:

You can use some assembler directives. Also known as pseudo op-codes.

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 the default.
DUMP address	The object code will be assembled at this address. 'Address' has to be a value from 16384 to 65535.
DUMP page,offset	Same as above, but allows code to be 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 the 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 allows you to include data. The data is followed after the directive.
DEFM "string"	Define message. This directive allows you to include text in ASCII form in the object code. Note. Don't use quotes inside quotes !!!
DEFS number	Define storage bytes. This directive allows you to create a gap of 'number' bytes. 'number' has to be a value from 1 to 16383. Used for buffers, tables etc.
DEFW numb,..numb	Define word (0-65535). Like DEFB, but for numbers from 0 - 65535.
label:EQU number	Equal or equate. This directive can only be used with a label. It allows you to give a value to that label.
LIST ON/OFF	This directive can be used to print a part or the complete source to the printer during assembly. The line printed will start with the current instruction address followed with the number of instruction bytes and the source line. The numbers are printed in hex. Directives do not have instruction bytes. When assembly has completed, Remove this directive from the source. This might save you troubles if you assemble again later on.

MDAT "file name"	This directive will merge a code file (or SCREEN$ file) from disk into the object code. There is no limit for the length of this file.
INC "file name"	Include source file "file name". This directive allows a source file which is on the disk to be assembled at the position of the directive in the main source file. INCludes cannot be nested - No INCludes are allowed inside a INClude file. The INClude file must be less than 24k long. The include files are loaded into the screen memory so if you see some strange things on the screen during assembly - don't worry - this is normal!

Note. both MDAT and INC work only with disks or RAMdisks (as used by MasterDOS).

There 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, (IX+dd), (IY+dd) etc.

The directives DEFB, DEFM, DEFS and DEFW may also be entered as DB, DM, DS and DW, but 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.
```

for example :
the instuction 'LD   IXL,123' becomes:

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

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

```
label:      EQU  $
```

or without the EQU $:

```
label:
```

NUMBERS

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

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 length.
```

To assemble, enter the command 'A' in command mode (SYMBOL C). During assembly the screen is deactivated. The screen will go black and the Z80 runs on the full 6 MHz.

When assembly is complete, you will see the same title screen as when you first entered COMET.
In the middle of the screen is some extra information : when assembly is completed succesfully, the message 'assembly completed' is printed. The message 'object code xxxxx Bytes' tells how many bytes were assembled. If more than 65535 bytes were assembled, then this number is the MODulus 65536 of the real number. If no multiple DUMP's were used (which can cause gaps), then this is also the length of the object code.
Finally, there is the message 'XXXXX Labels used' which tells you how many labels were defined during assembly.

If the message 'Assembling aborted' apears then there has been at least one error found in the source file.

When there are errors in the source, you can use SYMBOL N to find them. When you correct an error, don't forget to remove the error marker, otherwise you might wonder what's wrong with that line later on!!!

An error looks like this :

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

There are no limits to how much object code or symbols can be stored as long as there is enough memory free to store them.

* * E R R O R S * *

EDITOR ERRORS

Out of memory	There is not enough free 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 a block.
Number out of range	Number is above 255 or 65535.
String too long	String has more than 14 characters or more then 15 numbers are used.
Not found	Item was not found.
Invalid Workspace	Start bigger than end, or memory pages are already used.

ASSEMBLER ERRORS

The following errors are displayed on the screen if assembly has aborted.

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

SOURCE ERRORS

If an error is found in the source during assembly, then an error marker is inserted in the source file. If an error occured during INCluding, the errors which occured are placed above the relevant INClude statement in the main source file.

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 is above 255 or 65535.
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.
Wrong file type	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 another INClude file.

WARNING!!! :

Do not press the break button when you are in the editor or during assembly - the pointers will not be updated on re-entry and may be fatal for your source file. However, you may safely press the break button during execution of your object code.

SOURCE FILE CONVERTERS

On the disk are some file converters which can be used to convert sources from other SAM assemblers into a COMET source.

SC CONVERTER

If have used the SC_Assembler or have source files from it, you can use the program 'SC convert' to convert it to a COMET file. Please note that undocumented instructions are converted to DEFB 221/251 followed by the HL instruction.
Those SC Assembler commands which start with a asterix will be 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 own line.
The token ENT will be converted into a question mark, as this instruction has no use in COMET.

* * M A I N   M E N U * *

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-hand side are two small windows. The upper one holds the COMET version number. The window beneath it displays the current device, Which filename is used, the length of the source file in memory, the length of the symbol table and how much memory is still free in the workspace.

The bottom window holds the copyright message.
The window at the middle of the screen displays all possible options which are accessed by pressing the corresponding keys.

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 the extension '.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 filename is shorter than 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 file - Like load and merge, a filename extension '.S' will be added if the filename is less than 9 characters long.
If the file already exists, then the file on disk will be renamed as a backup version by using the extention '.B'. If there was aready a backup version on the disk, then that one will be overwritten. Any swap markers in the source (see SWAP) will be removed before the source file is saved.

N new device, enter the device character followed by the device number (if any). Devices may be Tape, Disk, or if MasterDOS was BOOTed also Network or ramdisk.

O save object code, if the file name is less than 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 source file editor.

* * I N S T A L L A T I O N * *

On the COMET disk is a program called 'COMET inst' which allows you to make some alterations in COMET to suit your preferances. These changes will be inserted into the machine code of COMET. When the 'COMET inst' file has loaded and is running, you will be asked several questions to answer.

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 font.	This option allows you to load in your own font from disk.

When you choose C or O, you will have to wait for a while as font is POKEd in an expanded version into 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, press 'Y' at this option 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

If your printer needs linefeeds after carraige returns, then press 'Y'
Default: Linefeed after carraige return

PRINTER CONTROL CODES

This option allows you to define a string of control codes to be sent to the printer each time you use the 'P' command.
This string can also be changed within the editor by using the 'I' command.
Enter your control codes as decimal numbers seperated by commas Up to a maximum of 14 numbers.
Default: undefined.

COLOURS

If you wish to have different colours on screen, then this option will allow you to change them.
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 on your disk.
Default: col 0:0,0 col 1:17,85(Flash) col 2:34,34 col 3:127,127

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 (see Workspace).

LOADING COMET

Insert the COMET disk and press F9. When COMET has loaded, it will be relocated to the first free page at the end of memory.
The workspace will be allocated to 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, here are some short demo sources. Have a look at them - I hope that you will find them useful in some way.

The programmer,
Edwin Blink.

* * E X A M P L E S * *

MDAT demomstration.

To demonstrate the MDAT (Merge DATa) instruction, this source will merge a screen into the object code.
When this code is executed, the screen will be displayed until 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
```

INClude create example.

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

This include file holds the subroutine '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
                 ADD  L              top 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  $

                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 in 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 if not
                RES   5,(HL)               reset key
                RET
                INC   "square I.S"         include.
                                           CALL routines: drawsquare
                                                          drawline
```

Include demo square2. Includes file "square l.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
                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 l.S"       include.
                                        CALL routines: drawsquare
                                                       drawline
message:        DEFM "COMET"
messend:
```

* * I N D E X * *