Manual for

SC_ASSEMBLER

512K and 256K

and for

SC_SPECLONE

SAM COUPE SOFTWARE

by

S.J. NUTTING

Steve's Software
7 Narrow Close, Histon, Cambridge, CB4 4XX
☎ 0223 235150 6pm - 9pm

NOTICE

[This pdf doc by Steve Parry-Thomas]
www.samcoupe-pro-dos.co.uk
[26 Jan 2005]
[For Sam Coupe Users Everywhere]

CONTENTS

INTRODUCTION

Like many users, I graduated to MGT's Sam Coupe from the Sinclair Spectrum. I had six years experience of writing machine code for the Spectrum, and since 1988 had been selling software for the Spectrum and PlusD disc drive - titles such as **Plus D Hacker**, **Plus D Toolkit**, and **Plus D Filer**. I had used five Spectrum Assemblers at one time or another, but when I came to program for the Sam Coupe with its vast memory, I found that none of them could handle enough source and object code. Writing large programs in 8K blocks of source code, assembling them and linking the object code files together, was just too complicated a process.

So my first Sam project was to write this big capacity editor/assembler. The 256K version of **SC_Assembler** accepts 10,000 lines of source code - enough to assemble 20K of object code. The 512K version accepts a massive 288K of source code which will assemble almost 64K of machine code. A Monitor enhancement is in preparation for the 512K version.

I have deliberately set out to make **SC_Assembler** user friendly, simple and practical. After all, I use it myself! It has a disassembler, facilities for inspecting Sam's DOS and two ROMs, and it accepts the undocumented codes which work on the Z80 chip but are not officially described by Zilog. **SC_Assembler** is easy to use, but sophisticated in operation.

The free utility **SC_Speclone** goes beyond the official Sam Emulator in achieving compatibility with 48K Spectrum programs. The Emulator can handle programs which make no, or few, Spectrum ROM calls, but it could never handle every possible ROM call unless the Emulator were a Spectrum clone - a breach of copyright which would not have gone unnoticed by Amstrad! Business and utility programs tend to use numerous ROM calls in order to leave the maximum memory space free for user's files.

SC_Speclone uses a copy of the Spectrum ROM, modified so that it will recognise the extra Sam keys like DELETE and the function keys. It enables you to switch between Spectrum and Sam modes, load Plus D snapshots, load tape software in Spectrum mode and save the whole Spectrum memory to Sam disc. You can print through Sam's printer port from Spectrum mode. For legal reasons, you must make the copy of the Spectrum ROM using your own Spectrum.

I hope you will enjoy using the programs. Any suggestions you have for future improvements and enhancements will be welcome. **S.J.N.**

All instructions for SC_Assembler apply to both the 512K and 256K versions of the program, unless otherwise stated.

MAKING WORKING BACKUP COPIES

The master disc is valuable, and you should not risk damaging or corrupting it by using it as a working copy. Write protect it, make backup copies of the programs for everyday use, and then keep the master disc in a safe place.

First, you need to prepare two newly formatted discs and copy the SAMDOS system file onto each of them.

You will also need to make a copy of your Spectrum 48K ROM for use with **SC_Speclone.** You must use your own Spectrum for this. It is illegal either to get a copy of the ROM from a friend, or to borrow a Spectrum in order to **make** a copy. Only those who have bought a Spectrum have the right to copy the ROM in this way, and then only for their own use.

Use a 48K Spectrum, or if you have a 128K model, put it into 48K mode. Even if you have a disc drive or microdrive, you must make the copy onto tape, or you will save the drive operating system, not the ROM.

```
SAVE "specrom" CODE 0,16384.
```

Boot up your Sam and load the DOS. Place the master disc in drive 1 and enter

```
LOAD 1
```

The first operation is to choose which of 32 different 64 column character sets suits you best. This is a very individual choice - some people will find one type of font easier to read, others will prefer another. The ENTER key will allow you to view the fonts available. When you return to the main instruction screen, enter the font number of your choice, or press ENTER again to re-view the fonts.

You will then be asked to specify the colours you prefer for PAPER, INK and cursor colour. Your choices will be demonstrated, and you should confirm that they are acceptable, or choose other options.

You should then specify whether you prefer to work normally in decimal or hex.

The choices of colour and numerical convention should be your usual preferences. It will be possible to change any of them by using POKES (see Appendix 2), and there are also commands to change hex/decimal output.

Finally, you will be asked whether you wish to have a line feed sent with every carriage return to your printer. If your printer is set to send a line feed automatically, reply "N". if your printer would always print on the same line if the line feed is not sent from within a program, reply "Y".

Now follow the screen prompts which will tell you when to put the master disc or your own disc in drive 1 to save the copy of **SC_Assembler.**

After **SC_Assembler** has been copied to your working disc, the program will go on automatically to make a backup copy of **SC_Speclone.** You should again follow the screen prompts, playing the Spectrum ROM tape as instructed, but use your second disc for making the copy. Do not put both programs on one disc because both boot up Sam, load the DOS and auto-run when function key 9 is pressed.

512K Sam owners will find that there is a copy of the 256K version of **SC_Assembler** on their master disc. There can be advantages in using the 256K version on a 512K machine when writing shorter programs, particularly if you are likely to want to assemble code to memory pages 16 to 28, which are not used by the 256K version. To make a backup copy of this version as well you will need a third formatted disc with a copy of the system file on it. Enter

LOAD 'STARTUP-256"

and follow the same procedure as for backing up the 512K version.

SAMPLE FILES ON 512K MASTER DISC

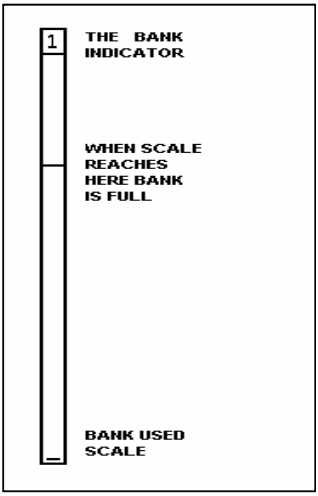
512K users are provided with two **SC_Assembler** files on the master disc, which may be loaded into **SC_Assembler** and printed out for reference.

Opcodes is a file of all the opcodes accepted by the program, including the undocumented codes.

Routines is a file of useful machine code routines which you may use in your own programs. They include a 64-col. screen printing routine, screen scrolling, keyscan and others. The file includes a full description of each routine.

THE EDITOR

Reset Sam, place your working disc in drive 1 and press key F9. On loading, you will be at the Editor working screen. On the right of the screen is the bank indicator.



The memory map in Appendix 2 shows in detail how Sam's memory is used by **SC.Assembler,** but briefly, source code is entered into memory banks holding 32K per bank The bank indicator shows the number of the current bank at the top, and the proportions of the current bank used and free at the bottom. As source code is entered, the scale grows upward, and when it reaches the "full" line you must select the next bank. 512K version has 9 banks, 256K has 3.

To change banks
Key F9

Increases bank number (When last bank reached, a further press selects bank 1)

Key F6
Decreases bank number (When bank 1 reached, a further press selects last bank)

Bn1 (as B3a/ENTER)
Switches to the bank number given, clears the screen and lists the first page of source code in the bank.

Bn1 I (as B3a/ENTER)
Switches to the bank number given without clearing the screen.

The same line numbers may be present in more than one bank, and Editor commands affect only the current bank. It can be helpful in programming to divide the various elements between different banks - the program main structure routines in one bank, subroutines and library routines in another, variables and data in a third. 512K users may also assign banks to screens and graphics.

Cursor

The cursor, a short line in your chosen colour, is at the top left of the screen.

Command syntax
The program commands are very short, mostly one or two characters. They many be entered in upper or lower case and should be followed by ENTER. A command written out in full (as list instead of 1), will generate the error message
missing number

Exit to BASIC
b or **q** or **x**

Line numbers
Like BASIC programs, each source code line must have a line number, which may be between 0 and 65534. Numbers may be typed in, or produced automatically by the program.

Auto line numbers
i
The program will generate automatic line numbers starting at 10 in steps of 10. (as 00010, 00020, 00030 etc.)

i nl (as **i** 2000)
The program will generate automatic line numbers starting at the number given, and in steps of 10 (as 02000, 02010, 02020 etc.)

i n1 n2 (as i 100 5)
The program will generate automatic line numbers starting at the first number given and in steps of the second number given (as 00100, 00105, 00110 etc.)

To quit autoline mode
SHIFT/ESC or SS/ESC

Renumbering
r
Renumber from start in steps of 10, making the first line number 00010

r n1 (as **r 100)**
Renumber from start in steps of 10, making first line number the number given (as 00010 00020 becomes 00100 00110 etc)

r n1 n2 (as **r 5000 5)**
Renumber from start in steps of n2, making the first line number n1 (as 00010 00020 becomes 05000 05005 etc)

r nl n2 n3 (as **r 30 400 2**)
Renumber from line n1, in steps of n3, numbering the lines from n2 (as 00010 00020 00030 00040 becomes 00010 00020 00400 00402 etc)

Line deletion
nl (as d 20)
Delete a single line (as 00010 00020 00030 becomes 00010 00030 etc)

d n1 n2 (as d 20 40)
Delete the block of lines from n1 to n2 inclusive (as 00010 00020) 00030 00040 00050 becomes 00010 00050 etc)

If **d** is entered alone, or if any of the numbers following it do not exist in the source code listing, the following error message is generated
missing number

Entering source code
Lines may be command or comment lines. A **command line** will have a line number, may include a label, and will include a standard Z80 opcode or one of the undocumented codes and the necessary operands. No remarks or notes may be included in a command line.

A comment line will have a line number followed by a semicolon ; Lines with only a semicolon may be used to break up the blocks of code for clarity, or any notes or comments may be included on the line, but it must not exceed 62 characters. A line which attempts to flow over into the next screen line will generate the error message
text line too long

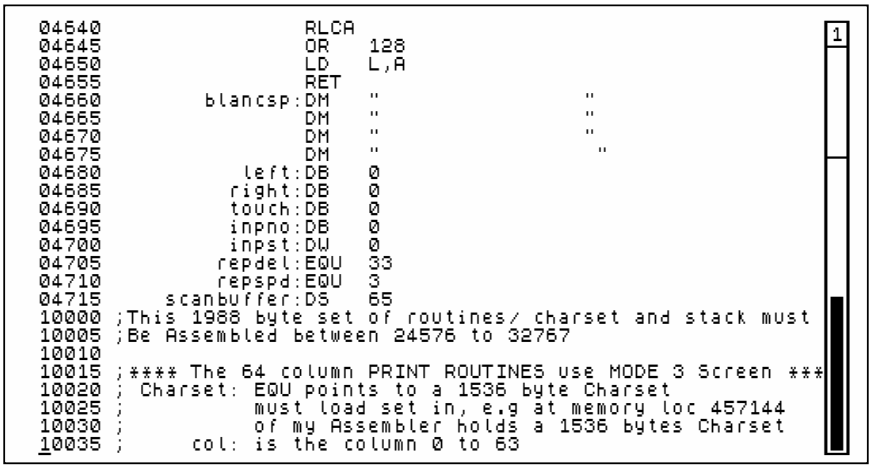
Labels
A label must not exceed 14 characters long, must not begin with a digit, and must be separated from the opcodes by a colon : A label must not stand alone on a line, but must be followed by opcodes The following characters may not be used in labels

+ - *: ; , ( ) % # “

The following error messages are generated by invalid labels
label too long (14 or less)
bad label (don't include +-*:;,()%#")
no opcode after label
Labels should not be duplicated, even in different banks.

Upper/lower case. Screen fields
Source code lines may be entered in upper or lower case and without

regard to tabbing to particular screen areas or fields. The program will adjust the tabbing automatically, to place labels and opcodes in the correct positions on screen for neat listing, and will change the case as follows:
Labels: changed to lower case
Opcodes and operands: changed to upper case
Comments: Unchanged - left exactly as you typed them.



The illustration shows the screen display of source code listing, including comment lines, a command line with a label, and command lines without labels.

Cursor movement
Arrow keys
The arrow keys move the cursor as you would expect. The cursor stops moving when the right or left keys bring it to the screen edge. When using the vertical movement keys, the screen scrolls to bring in another line when the cursor reaches the top or bottom of the screen.

Quick cursor movement
Key F2
Moves the cursor to the end of the previous word to the left

Key F3
Moves the cursor to the beginning of the next word to the right

Key F0
Scrolls down 22 lines. Previous bottom line is moved to two lines from the top

Key F1
Scrolls up 22 lines. Previous top line is moved to 2 lines from the bottom

EDIT
Move cursor to top left corner of screen

SHIFT/EDIT or SS/EDIT
Move cursor to bottom left corner of screen

n n1(as **n** 2570)
Display 24 lines on screen, with line n1 at the top

e n1 (as **e** 200)
Display line n1 at the cursor position.

Other Editor keys/commands
Key F5
Insert a blank line at the cursor position for writing on.
Additional source code lines need not be inserted into the listing in the right place, provided they are given the correct line number. When the source code is listed again, the line will have been placed in the correct sequence. A blank line may also be used for entering other Editor commands.

TAB
Toggle insert/overwrite mode.
A space is inserted at the cursor position and new characters may be entered without overwriting those already there. The line will be adjusted to accommodate them. If inserted letters cause the line to overflow, characters will be lost from the right hand end Pressing **TAB** a second time closes up the remaining extra space and reverts to overwrite mode.

DELETE
Delete character at cursor position leaving a space at cursor position

SS/DELETE
Delete character to right of cursor position and close up gap.

ESC
Clear screen

CAPS
Toggle caps lock on and off

In practice, caps lock is only useful in comment lines, because the case is adjusted automatically in command lines.

Key F8
Delete the line at the cursor position from the screen display and scroll the lines below up one line. NB. The line is only deleted from screen not from memory. If the file is listed again, the line will still be present.

l list source code
List the source code to screen, starting from the first line and scrolling continuously.

1 n1 (as **l 100)**
List the source code to screen, starting from line n1 and scrolling continuously. If n1 is a non-existent line number listing will be from the next valid line number. If n1 is greater than the last line number, the following error message is generated
number not found

l n1 n2 (as **1 100 500)**
List the source code block from line n1 to line n2 to screen, scrolling continuously. If n2 is greater than n1 the following error message is generated
end bigger than start

During screen listing, any key toggles scrolling pause, and ESC aborts listing.

11 list to printer
List source code to printer
Line numbers may be added as for command 1 above.

SHIFT/KEY FO or SS/KEY FO
List the first 24 lines of the source file to screen without scrolling.

SHIFT/KEY F1 or SS/KEY Fl
List the last 24 lines of the source file to screen.

f find
Find strings or numbers in source file. Will not search for opcodes. Searches all banks, and lists the bank number and each line number in which the object of the search is found.

To edit a line displayed by the search
Place the cursor over the line's bank number and press ENTER The lines in that bank may now be edited.

f label
Find label at place at which it is initialised.
NB. If the string of letters appears in a longer label, that too will be listed. (as **f end** would find the labels **blend end endgame** etc.)

f:label
Find all calls and jumps to a label.
Same string conditions as above. (as **f:end** would find and list **DJNZ end JR NZ blend** etc.)

f n1
Find whole decimal number

f #n1
Find whole hex number

f %n1
Find whole binary number

f “1
Find a single letter variable (as CP "1" or LD A,"1")

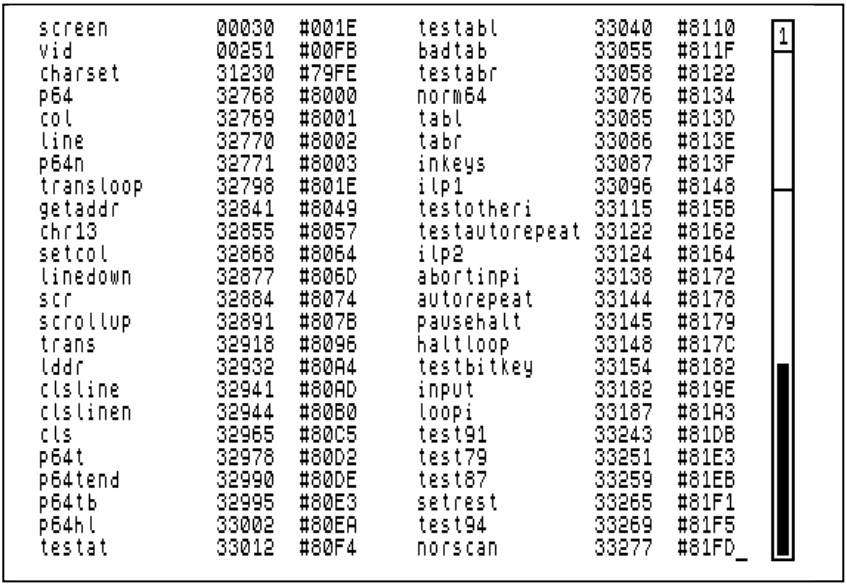
f "text
If two or more characters are listed after the inverted commas, only the bytes after **DM** opcodes will be searched.

f ;text
Find the given string in comments lines only.

Symbol table
The following commands are available only after assembly.

s
List all labels to screen
List labels in order in which they are initialised in the program, giving hex and decimal values. Pauses when screen full Any key clears screen and lists next screenful.

sl
List all labels beginning with the letter specified. The labels will be listed in the order in which they were initialised in the program, but this command allows labels to be listed in rough alphabetical order by repeating the command. (as **sa sb sc** etc).



The illustration shows the screen display of the symbol table

s: or s:l
As above, but list labels to printer.

The s commands will list the symbols found in all banks.

ESC
Aborts symbol table listing

Number convention and conversion
+d
All future output to screen or printer to be decimal listing

+h
All future output to screen or printer to be hex listing

+dnl
Display decimal number with its hex and binary equivalents

+hnl (without #)
Display hex number with its decimal and binary equivalents

+bn1 (without %)
Display binary number with its decimal and hex equivalents.

Hex numbers may only include the digits 0-9 and A-F. Any other characters will generate the error message
hex error (0-9 A-F)

Binary numbers must always have eight digits. Any fewer will generate the error message
too few bin (8 0's or 1's)

Decimal, hex and binary numbers may be entered in source code listing. Numbers with no prefix will be taken to be decimal numbers. Hex numbers must always be prefixed by **#**
Binary numbers must always be prefixed by **%**

Specially defined keys.
A number of keys have been predefined to give characters which are in frequent use when writing source code.
F7 gives **(**
F4 gives **)**
INV gives **#**
CNTRL switches the Function keys to keypad mode. In this mode Keys F0-F9 give the digits 0-9, and the full stop is converted to a comma. This is to facilitate the entering of lists of data bytes separated by a comma. While in keypad mode **SHIFT** or **SS** plus a function key will give its normal use as described previously. Pressing **CNTRL** while in keypad mode returns the function keys to their normal uses.

Opcodes
798 opcodes are recognised. They must be entered in the normally accepted form In addition to error messages described previously, the following syntax errors will generate error messages
LD HL, or **LD A,**
variable missing
LD A,256
number too big (0-255)
LD A,+7 or **LD HL,7-** or **DB 7++5**
+ **or - in wrong place**
DB 40, or **LD** A**,(7** or **LD A,)7)**
() expected
DB "**ap**" or **CP "a**
letter bad/quotes expected
RST 100 or **RST #49**
rst bad only 8 are allowed
DJNZ 45
DJNZ must start with a label
10 "text"
missing DM
10 128
missing DB/DW/DS

When an error message is generated, use the **UP arrow key.** This will delete the message and leave the cursor at the line to be corrected.

Take special care with the spacing of the following codes

CPD	CP D
CPL	CP L
RLA	RL A
RLCA	RLC A
RLD	RL D
RRA	RR A
RRCA	ARC A
RRD	RR D

Undocumented opcodes

There are 102 undocumented codes, most of which operate on the upper or lower parts of the IX and IY registers. Typical syntax is

LD IXl,50 (load the lower element of the IX register with 50)

CP IYh Compare contents of high element of IY register with contents of A register.

Pseudo-opcodes

The following pseudo-opcodes are recognised

DB (not **DEFB)**

Define an 8-bit byte (as **DB 128, #C9, %00000001, label)**

DW (not **DEFW)**

Define a 16 bit number to be held in consecutive bytes, with the LSB in the first byte, the MSB in the second. (as **DW 32768, #C000, label).** When a number lower than 256 follows a DW instruction, the first byte will hold the number and the second will hold 0.

DM (not DEFM)

Define message text (as **DM "message")**

DS (not **DEFS)**

Define spaces (as **DS 256** or **DS label)**

EQU

Define a label as an 8 or 16 bit value (as **start:EQU 49152)**

ORG

Define the address from which the code is to be run. **ORG** may be to any address between 0 and 65535. If no **ORG** command is included, the default address is 32768.

PUT

Define the address at which the object code is to be stored during assembly. **PUT** may be to any address between 16384 and 65535. If no **PUT** command is included the default address is 32768.

THE ASSEMBLER

a

This command invokes the two-pass assembler. The code in all banks is assembled in bank order, starting with bank I. The line numbers in each bank are only relevant in that each bank's source code is assembled in line number order, but if the lines in bank I have higher numbers than those in bank 2, they will still be assembled first.

Pass 1 tests for 5 serious errors, and the following error messages may be generated.

PUT too low

PUT must not be below 32768.

PUT past >65535

The PUT value was too high for the size of code block being assembled, and caused it to be poked beyond 65535.

label not found

A label referred to has not been defined (as **loop:EQU start** or **DS start** when **start** has not been defined).

label defined twice

The same label name may not be used more than once, even in different banks.

out of label space

The maximum number of labels permitted is 2048 in 512K version, 1024 in 256K version.

If an error is found, the assembly is aborted and the line or lines containing the error displayed on screen for correction.

If pass 1 is completed successfully, the following information is displayed on screen:
ORG start and end addresses and code block length in decimal and hex PUT start and end addresses and code block length in decimal and hex Number of labels used.

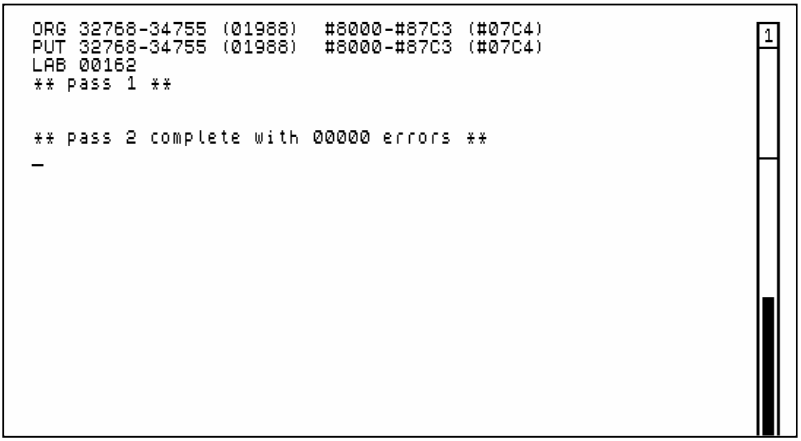
Pass 2 tests for 2 possible errors, and generates the following error messages

JR/DJNZ out of range

JR and **DJNZ** offset must be between -126 and +129 bytes. When this error is generated, change **JR** instructions to **JP** and **DJNZ** to **DEC B JP NZ.**

An error message will abort the assembly and display the line containing the error on screen for correction.

If pass 2 is completed successfully, the illustration below shows the screen display.



Assembler commands
The following assembler commands may be entered as lines in the source code listing.

***report off**
The assembly will no longer be halted by an error, but will be completed. The total number of errors found will be shown in the final display, and all the error lines displayed together.

***report on**
Reverts to the normal status, so that assembly will be halted by an error.

***list on**
During assembly, from the line which contains this instruction, all object code bytes, their running addresses, source code line numbers and source code lines will be listed to screen. The hex/decimal convention will be determined by the choice you made when setting up the working copy of the program, but may be changed by the direct commands **+d** or **+h.**

***list off**
Reverts to assembly without listing from the line containing this instruction.

***printer on**
Similar to *list on, but listing is sent to printer. The printer must be set to more that 80 columns, by using elite or condensed type and setting the right margin at column 96 or more.

***printer off**
Stops listing to printer from the line containing the instruction.

During assembly with listing, any key toggles pause. ESC aborts the listing and the assembly. Listing may be to screen and/or printer, but assembly without any listing is much faster.

***print**
Prints a label value, message, or reminder to screen during assembly. (as ***print start** which will give the value of the label start in hex and decimal at the end of pass 1, or ***print** "**version 2**" which will print the text message when the line containing the instruction is reached in pass 2)

cn1 Bank paging
(as c19 entered as a direct command)
c, followed by a valid page number, causes the bank starting with the page specified to be paged to the address 32768 and the object code will be assembled to that bank. The new paging remains until another c command changes it.

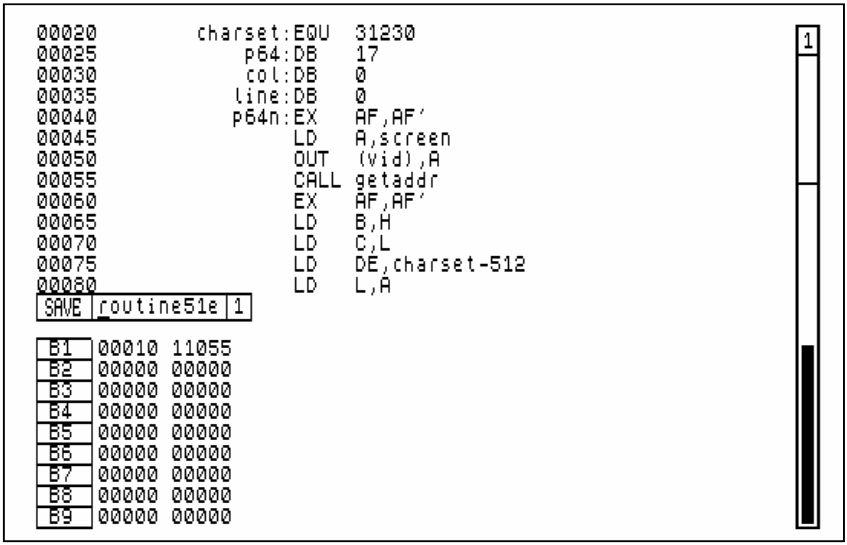
Banks normally reserved for source code may be used but there must be no source code in the bank selected for assembly, and the PUT address must then be specified as **PUT 32770** or higher, because the first two bytes of source banks are markers holding 255,255 and must not be overwritten. The normal default address of 32768 would cause the object code to overwrite the markers. In the 512K version c3 may be used to bring the EXTRA memory, reserved for assembling code, into action and this does not require the higher PUT address. See the memory maps in Appendix 2 for the page numbers which may be used.

If using the 256K version on the 512K machine, the pages not normally available on a 256K machine may be used in this way.

c1
Reverts to normal paging.

THE DOS COMMANDS

+s
Save source code file



The illustration shows the screen when the **+s** command is entered. The SAVE information is displayed at the bottom left of the screen.

In the box next to SAVE is the file name, which can be changed. Move the cursor to the first letter of the file name, and pressing any character will clear the box. Type in the new name. The DELETE key is inoperative, so use the left arrow key to move the cursor back and overwrite any errors.

To the right of the filename is the drive number. If this is to be changed use the down arrow key to move the cursor to the drive number box and type in the drive number.

The display of source code numbers found in each bank is at the bottom of the screen. Press **ENTER** to save all the source code in all of the banks.

To save part of the source code file, change the line numbers in any bank containing blocks to be saved to the first and last lines required in that bank. Change the line numbers in all banks which are not to be saved to **00000 00000**. Use the arrow keys to move the cursor. The right and left arrows move it only within each five-figure number. The down and up arrows move to the next or previous number.

ENTER
save the source code blocks specified

SAVE errors
If the cursor is returned to the bank numbers display without saving the files, an error has occurred. Either the lines specified for saving do not exist, or a start number higher than an end number has been given. Correct the error and press ENTER again.

The files saved will be a header file, plus one file for each bank saved.

+l
Load source code file

A box with LOAD, the filename and drive number is displayed. Follow the same instructions for changing these as given for the **+s** command. **ENTER** to load the file. All source code files already in memory will be cleared and the new file loaded into the banks from which it was saved.

+m
merge source code files

The file name and drive number must be specified as usual, and the file will then be loaded into memory and merged with the existing file. Two errors are possible, generating the following error messages

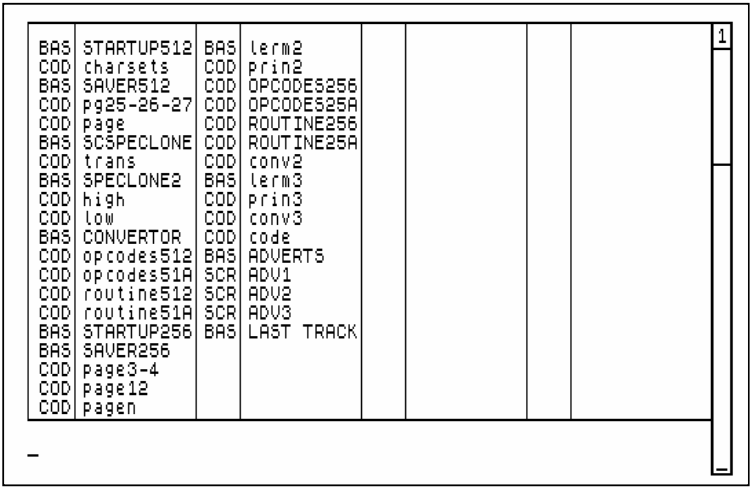
Not enough room to merge BANK n1 Press any key
The file being loaded and merged contains too many lines in the bank number given to be merged with the file already in memory. The merge cannot be done unless part of the file in memory can be deleted.
Same lines exist in BANK n1 Press any Key
The file being merged has the same line numbers in the bank number given as those used in that bank by the file already in memory. Renumber the source code lines in the file already in memory in that bank, and enter +m again, because the merge operation will not overwrite any existing lines.

+e
Erase an SC Assembler file from disc
Give the filename and drive number as usual, and all the source code files will be erased from the disc. Use only for **SC_Assembler** files.

ESC
In all disc operations, pressing ESC instead of ENTER will exit the mode without performing the disc operation.

+c
Catalogue the disc

The illustration shows the disc catalogue screen.



Only Sam files are shown in the catalogue. Any Spectrum files an the disc appear as gaps in the listing. The file type is displayed beside each filename, and any Sam files which have been erased and not yet overwritten appear with ERA in the file type column. The display remains on screen if any other disc operation is then selected.

Saving object code
Object code must be saved from BASIC. After assembly, note the PUT address and file length displayed after pass 1. The paging set by the c command should be left as it was when the code was assembled.

Exit to BASIC and enter
SAVE "filename" CODE start,length
Use the filename of your choice and the start address and file length noted from the assembler display.

Use Key F4 to re-enter the assembler. The source code file in memory is preserved, and the bank paged to 32768 is unchanged.

Loading object code files
Object code files may be loaded from BASIC. If screens or other code files are to be loaded into unused source code pages, ensure that the load address will not overwrite the first two bytes of the bank.

THE DISASSEMBLER

+z
Enter the disassembler

The disassembler has two modes, NORMAL and ROMS. On entering the disassembler, the screen clears and a status box is displayed at the bottom of the screen. Beside DISS is the address from which disassembly is to take place, and will show 32768 on entry. This may be changed by typing in any address between 0 and 65535, (*0 and #FFFF hex).

To the right of the address is the MODE. The down arrow key moves the cursor to this box, and the up arrow may be used to toggle between modes.

On the extreme right is the box showing printer status. Use the down arrow key to move the cursor to this box, and the up arrow to toggle ON/OFF. When OFF shows, the disassembly output will be sent to the screen. When ON shows, disassembly listings will be sent to the printer.

NORM mode
Memory 0-65535 is arranged as follows

00000-16383 ROM 0
16384-65535 Normal RAM as in Sam BASIC.

ROM 0 can be disassembled in this mode, and any other code files may be loaded from BASIC to addresses between 16384 and 65535. Code files may also be loaded into addresses in unused source code banks provided that the first two bytes of the bank are not overwritten. The command c is used to page the bank containing code to be inspected to address 32768.

ROMS mode
Memory 0-65535 is arranged as follows

00000-16383 ROM 0
16384-32767 DOS
32768-49151 Normal RAM as in Sam
BASIC 49152-65535 ROM 1

This mode makes it easy to inspect Sam's two ROMs, and the DOS Other code files may be loaded to 32768, but only one page is available for normal RAM in this mode.

ENTER
Starts disassembly listing
All listing will be in hex or decimal according to your choice when the program was set up, unless **+d** or **+h** have been used to change the convention. The Disasmbler will not recognise blocks of data bytes, and will interpret them as instructions.

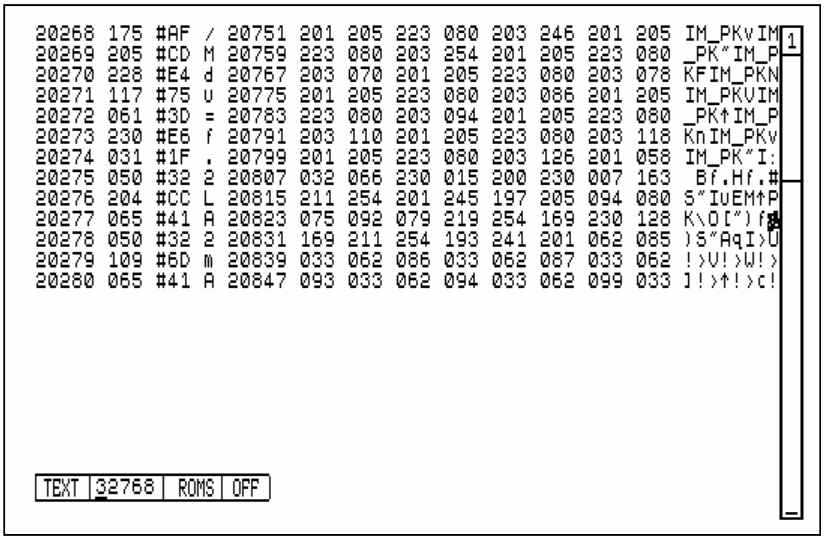
Any key except SPACE or ESC
Pauses listing.
Press key again to continue listing

SPACE
Abort listing
Stays in disassembly mode and returns to the status box for change of address, mode or printer status.

ESC
Abort listing and exit disassembler.
Returns to Editor mode.

TEXT LISTING

+t
Enter text listing.



The illustration shows the text listing screen.

The same NORM and ROMS modes and printer status are available as with

as with the Disassembler, and they are changed in the same way. **ENTER, SPACE** and **ESC** operate as they do in the Disassembler. The text listing gives a double memory dump, to screen or printer. Decimal or hex listings will be determined as described for the Disassembler.

The four columns on the left of the text listing display list single addresses, the contents of each address in decimal and hex, and the ASCII equivalents for bytes between 32 and 127 decimal Bytes outside the ASCII printable character range are represented by a full stop.

The remaining columns list an address, its contents and the contents of the following 7 bytes in hex or decimal depending on which is the current mode, and the ASCII equivalents of the B bytes.

Although the two sides of the split screen start the listing at the same address, the right hand display moves much faster through memory than the left and the two displays are very soon showing different areas of memory. Use the right hand display to search quickly for large blocks of code, messages etc. and the left for detailed examination of sequences.

PRINT INFORMATION SCREEN
256K VERSION ONLY

+p

Call up information screen. [256K only]

The command clears the screen and calls up an information screen which lists the most commonly used Sam Coupe parts. Where the significance of a number read from or written to a port lies in the individual bits rather than the whole number, the use of each bit is explained.

[NOTE: This was crossed out in my manual so the command might not work. – SPT 22-Jan 2005]

The information screen is not available in the 512K version. The command **+p** is not rejected by the 512K program, but no operation is performed if it is entered.

TESTING ASSEMBLED CODE

After assembly, machine code routines may be CALLed from BASIC for testing, provided that the ORG and PUT addresses are the same. See Appendix 2 for information on the Intelligent Block Transfer which may be used if ORG and PUT are different. It is prudent to save both source and object code to disc before calling untested machine code, in case the program crashes.

CLEAR SOURCE CODE BANKS

512K
Exit to BASIC
Enter RUN
1100

256K
Exit to BASIC
Enter RUN 1010

The entire source code file will be cleared from all memory banks. Object code will not be erased, even if it has been assembled to a bank normally reserved for source code.

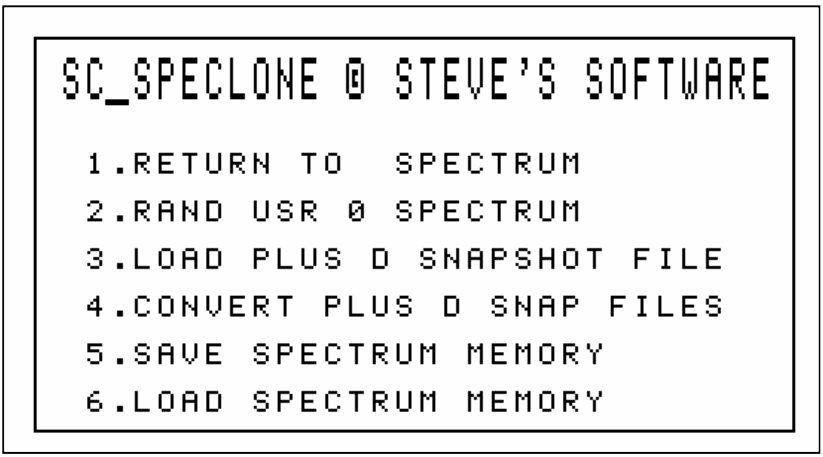
SC_SPECLONE

SC_Speclone is a bonus utility program supplied tree with **SC Assembler.** It allows a wide range of Spectrum 48K programs to run on Sam, even business and utility programs which use a printer.

Plus D disc snapshot files can be converted for use with Sam, and Spectrum 48K tape programs can be loaded and the whole Spectrum memory saved to disc to be re-loaded for future use. Only snapshots can be used from Plus D discs, not program files. So if you have a disc based Spectrum 48K program which you wish to use with Sam, and have no tape version, make a 48K snapshot using the Plus D. 128K programs and snapshots cannot be used with **SC_Speclone.**

Appendix 2 contains technical information about the working of this utility. On

loading, the program goes to the main menu, illustrated below.



Menu option 1
Return to Spectrum

Selecting this option will switch to Spectrum mode. If there is a Spectrum file in memory, this will be preserved.

In Spectrum mode
All keys are scanned, but the following Sam keys have serial uses:

TAB Graphics mode - equivalent CS/9
CNTRL E-mode - equivalent CS/SS
ESC BREAK - equivalent CS/SPACE

FUNCTION KEYS 0-9 Keypad returning the digits 0-9
INV hash mark #

Printing from Spectrum mode
LPRINT sends ASCII characters to the printer, with tokens unexpanded. For example, LPRINT 'Testing" will work **LliST** also sends only ASCII characters and unexpanded tokens, so it is not possible to LLIST a Spectrum program, because the keyword codes will not be converted to spell out the keywords.

Linefeeds
The program is set up to send a linefeed after each carriage return. If your printer already sends a linefeed automatically, the result will be permanent double line spacing. To correct this, the Spectrum ROM file on the **SC_Speclone** disc must be modified.
Reset Sam, after saving the Spectrum memory if necessary. Put the **SC_Speclone** disc in drive 1, and enter the following lines as direct commands

```
CLEAR 32167
LOAD "ram" CODE
POKE 80324,195
SAVE OVER "rom" CODE 65536,16364
```

Printing to #3
Most Spectrum utilities do not use LPRINT, but send the bytes to be printed to be output to *3 via a printer driver routine. **SC_Speclone** has a suitable printer driver at Spectrum addresses 14793-14826 (34 bytes) (Sam addresses 80329-80362). This routine should be copied to the address at which the Spectrum program's own printer driver routine resides, overwriting the program's routine.

Spectrum addresses from Sam BASIC
From Sam BASIC, all Spectrum addresses reside 65536 bytes higher than their normal Spectrum address. (See memory map in Appendix 2).

Switching to Sam from Spectrum mode
NEW
The Spectrum keyword NEW returns to Sam. If Spectrum address 23296 holds 0, the return will be to the main menu. If 23296 holds any other value, the return will be to Sam BASIC line 1000

NMI button.
If the Spectrum program does not permit you to exit to BASIC, the NMI

button can be used to return to Sam BASIC, but Sam's NMI button has a fault which needs a hardware modification. It will exit to Sam BASIC, but you should not rely upon being able to return to the Spectrum program at the point from which you left it.

To NEW the Spectrum memory.
The keyword NEW cannot be used in the usual way, because it is used to switch to SAM mode.

PRINT USR 14888
Mimics NEW. BASIC programs are cleared from memory, but code stored above a CLEAR address is preserved.

PRINT USR 0
Besets Spectrum, clearing all programs and code from memory.

Saving and loading from Spectrum mode
In Spectrum mode, SAVE and LOAD will normally be to tape, and the program compensates automatically for tape loading speed. Disc and microdrive syntax are not accepted. Disc saving and loading of code blocks must be done by storing variables such as file start and length in Spectrum memory, switching to Sam, retrieving the variables and saving the code block from Sam BASIC.

Example of conversion of Spectrum program for file saving to disc.
The Spectrum variable 23296, which controls the method of returning to Sam BASIC, will hold 0 if the return is to the main menu. If it holds any other number, the return will be to Sam line 1000. In Sam BASIC, Spectrum addresses lie 65536 above their working Spectrum address, and so the contents of this variable can be retrieved from Sam BASIC by PEEK(65536+23296).

A typical Spectrum microdrive SAVE routine is

```
5000 GOSUB 6000:SAVE * "M";D;F$ CODE start,length:RETURN
6000 LET D=PEEK 32768:LET start=PEEK 32769+256*PEEK
32770:LET length=PEEK 32771+256*PEEK 32772:LET F$="":FOR
A=0 TO 9:LET F$=F$+CHR$ PEEK(32773+A):NEXT A:RETURN
```

To convert this to saving from Sam, the Spectrum BASIC must be

```
5000 POKE 23296,1:NEW:RETURN
delete line 6000
```

23296 holding 1 will force the jump to Sam line 1000 when NEW switches to Sam. The Sam command GO TO 1 will return to Spectrum mode to execute the instruction immediately following NEW. 23296 could be made to hold 1 for SAVE, 2 for LOAD, 3 for DIR, 4 for ERASE and so on. The necessary Sam BASIC will be

```
1000 LET N=PEEK(65536+23296)
1010 IF N=1 THEN SAVER: ELSE IF N=2 THEN LOADER: ELSE IF N=3
THEN DIR:ELSE IF N=4 THEN ERASE:END IF
5000 LABEL SAVER: GOSUB 6000: SAVE "D"+CHR$(D+48)+":"+F$ CODE
start+65536,length:GO TO 1
6000 LET D=PEEK (32768+65536): LET start=DPEEK(32769+65536):LET
length=DPEEK(32771+65536):LET F$=MEM$(32773+65536 TO
32773+9+65536):Return
```

Line 1000 will fetch the contents of the Spectrum variable 23296, and line 1010 will call the appropriate Sam subroutine to perform the correct disc operation. 23296 holding 1 would call the save routine at line 5000. Line 6000 mimics the Spectrum line 6000, but peeks addresses 65536 above the Spectrum ones and uses Sam's more economical syntax. After calling line 6000, line 5000 saves the required code block to disc and GO TO 1 returns to Spectrum mode to execute the RETURN which follows NEW in Spectrum line 5000.

Similar subroutines could be written to perform the other disc operations. Note that Sam labels cannot be keywords such as SAVE or DiR - another letter must be added. See also Appendix 3.

Menu option 2
RANDOMIZE USR 0 SPECTRUM

This option returns to Spectrum mode and clears the Spectrum memory. It should be used the first time a jump is made to Spectrum mode, unless a Spectrum program has already been loaded.

Menu option 3
LOAD PLUS D SNAPSHOT FILE

This option should only be used to load Plus D snapshots which have previously been converted using option 4.

If, when a converted snapshot is loaded, there is no response to the keyboard on returning to Spectrum mode, use the NMI button to return to

the Sam mode menu and follow this procedure

Press ESC - goes to Sam BASIC.
Enter POKE 80290,195
Enter GO TO 10 - returns to the **SC_Speclone** menu
Use option 3 and re-load the snapshot.

Most snapshots are compatible with the program, and this POKE will enable the keyboard response for the majority. If you have a snapshot which requires this POKE, in future exit to Sam BASIC and do the *POKE* before loading the snapshot.

Menu option 4
CONVERT PLUS D SNAP FILES

This option must be used before a snapshot can be run under **SC_Speclone.** If the POKE described above has been used to enable the keyscan of another snapshot, it must be restored before using this option.

Press ESC to go to BASIC.
Enter POKE 80290,226
Enter GO TO 10 to return to menu.

Put the disc containing the Plus D snapshot into drive I and select option 4. The catalogue will be displayed, and you will be prompted for the file number of the program to be converted. After a brief pause while the conversion is made, you will be prompted for the filename under which it is to be saved. Put the disc on which you wish to save it in drive 1 and give the filename. After saving, the program returns to the main menu. Option 3 may be used to load the converted snapshot.

Menu option 5
SAVE SPECTRUM MEMORY

The complete Spectrum memory, from Spectrum addresses 0-65535 is saved to disc.

Menu option 6
LOAD SPECTRUM MEMORY

Loads files saved under option 5. After loading, select menu option 1 to return to Spectrum BASIC with the memory preserved.

APPENDIX 1
GLOSSARY OF TERMS USED

ASCII
American standard code for information interchange.
The standard used to ensure that computer programs all use the same codes for printable characters.

ASSEMBLE
Convert source code file to the block of code which is needed for a machine code program.

BANK
A block of memory assigned for a particular purpose. In **SC_Assembler** a bank is usually a block of 32768 bytes.

BINARY
System of arithmetical notation using only 2 digits - 0 and 1. If the binary form of a number is used with **SC_Assembler** it must be entered as 8 digits preceded by % as
%00000011 (binary equivalent of 3)

BIT
BInary digi**T**
There are 8 bits in a byte.

BYTE
Unit of computer memory. Each computer memory address can store one byte - usually a number between 0 and 255.

CURSOR
Marker used on screen to show the current printing position.

DATA
Figures in a computer program which are not to be interpreted as instructions. They may be variables, message character codes, printer codes etc.

DECIMAL
Familiar system of arithmetical notation using 10 digits 0-9.

DISASSEMBLE
Examine code block and convert the bytes to opcodes as used in source

code listing. Does not produce perfect source code because data bytes are not recognised and will be mistakenly converted to opcodes. Data byte sequences are usually easily recognisable because the sequence of opcodes is nonsensical.

DOS
Disc **O**perating **S**ystem
Computer system routines which control the disc drive. The routines will switch on the drive motors and move the drive heads as necessary and will perform operations such as SAVE, LOAD, CATALOGUE, ERASE, READ, WRITE, VERIFY.

HEX
System of numerical notation using 16 digits, the numbers 0-9 and the letters A-F, frequently used in computer programming. In Sam BASIC hex numbers are introduced by &, in **SC_Assembler** by # as
&C02F #C02F (hex equivalent of 49199)

LABEL
Distinctive name used to mark a particular point in a program and given a value which identifies that point. In source code and in Sam BASIC, reference to the point in the program may be made by referring to the label

MACHINE CODE
Computer program consisting only of the sequence of numbers needed to cause the computer to perform the required actions. Machine code programs are very fast in execution, because the computer spends no time interpreting the instructions.

OBJECT CODE
Block of bytes - the machine code program - produced by an assembler after converting source code.

OPERAND
Number which follows an opcode in source code listing.
In LD A,4 LD A, is the opcode, 4 the operand.

OPCODES
Standardized list of mnemonics used when writing machine code programs in source code.

PAGE
Block of 16384 bytes. The Z80 processor can address only 65536 bytes at

any one time, so Sam's memory is divided into pages, any 4 of which can be addressed by the processor at a time. The pages addressable by the processor are said to be 'paged in'. There are 32 pages in a 512K Sam, 16 pages in a 256K machine.

PSEUDO-OPCODES
Opcodes which instruct the assembler to perform some action and will not be converted to machine code instructions. They include
ORG - instructs the assembler to assemble the code to run from a particular address
DM (define message) - instructs the assembler to insert the ASCII codes for a text message.
EQU - instructs the assembler to assign a value to a label

RAM
Random **A**ccess **M**emory.
That part of memory which is available for variables, programs and graphics, and whose bytes may be changed by the user. The bytes can be PEEKed and POKEd.

ROM
Read **O**nly **M**emory.
That part of the memory which contains the computer operating system routines which cannot be changed by the user. The bytes may be PEEKed, but cannot be POKEd.

SOURCE CODE
Listing of a machine code program using opcodes to represent the instructions, so that the logic and operation of the program may be more easily followed. May include notes and instructions to the assembler as well as the opcodes which will be converted into program bytes. Must be 'assembled' - converted to a block of machine code bytes - before the program can be run.

WORD
Two consecutive bytes holding a number over 255, in the following form
For a number n
byte 1 (the least significant byte or LSB) holds n-(256*INT(n/256))
byte 2 (The most significant byte or MSB) holds INT(n/256)

APPENDIX 2
TECHNICAL INFORMATION

SC_ASSEMBLER
Memory maps and useful addresses to poke
The 512K version memory map

PAGE	Type	Memory loc.	
0	LOW	16384-32767	Normal Sam RAM 16384-16883 contains Assembler Code
1/2	HIGH	32768-65535	Normal Sam RAM Area reserved to assemble to
3/4	EXTRA	65536-9M3	Extra area reserved to assemble to
5/6	SOURCE1	98304-131071	32K source bank 1
7/8	SOURCE2	131072-162839	32K source bank 2
9/10	SOURCES	163840-1%607	32K source bank 3
11/12	SOURCE4	196608-229375	32K source bank 4
13/14	SOURCES	229376-262143	32K source bank 5
15/16	SOURCE6	262144-294911	32K source bank 6
17/18	SOURCE7	294912-327679	32K source bank 7
19/20	SOURCES	327680-360447	32K source bank 8
21/22	SOURCE9	360448·393215	32K source bank 9
23/24	LABEL	393216-425983	32K Symbol table label store 2048 labels max.
25/26/27	CODE	425984-475135	48K Assembler/Disassembler program code
28	MONITOR	475136-491519	16K Area reserved for future Monitor program upgrade
29	SYSTEMDO	491529-507903	16K System DOS
30/31	SCREEN	507904-0-16383	32K Normal Sam screen area

512K version keyscan variables
The program keyscan has variables which may be POKEd to change the keyscan response to suit personal preference.

442545 (equivalent to Sam SVAR 521 REPDEL) Holds number of 50ths of second delay before a key repeats - 33 normally.

442572 (equivalent to SAM SVAR 522 REPSPD) Holds number of 50ths of a second between repeats

The 256K version memory map

PAGE	Type	Memory loc.	
0	LOW	16384-32767	Normal Sam RAM 16384-16883 contains Assembler Code
1/2	HIGH	32768-65535	Normal Sam RAM Area reserved to assemble to
3/4	CODE	65536-98303	32K Assemble/Editor part 1
5/6	SOURCE1	98304-131071	32K source bank 1
7/8	SOURCE2	131072-162839	32K source bank 2
9/10	SOURCE3	163840-196607	32K source bank 3
11	LABEL	196608-212991	16K Symbol table label store 1024 labels max.
12	CODE	212992-229375	48K Assembler Editor code part 2
13	SYSTEMDOS	229376-245759	16K System DOS
14/15	SCREEN	245760-2621538	32K Normal Sam screen area
		0-16383	

256K version keyscan variables

The program keyscan has variables which may be POKEd to change the keyscan response to suit personal preference.

82097 (equivalent to Sam SVAR 521 REPDEL) Holds number of 50ths of second delay before a key repeats - 33 normally.

82124 (equivalent to Sam SVAR 527 REPSPD) Holds number of 50ths of a second between repeats.

Pokes to change Editor screen colours and hex/decimal listing - BOTH VERSIONS

16440 holds PAPER colour (0-127)
16441 holds INK (PEN) colour (0-127)
16442 holds 0 for decimal output, 1 for hex output
16443 is unused
16444 holds cursor colour (0-127)

These may be changed temporarily. Exit to BASIC and POKE the appropriate variables with the new values. If you wish to make the changes permanent, do the POKES and

SAVE OVER "page" CODE 16384,500

Intelligent block transfer

When source code has been ambled, the machine code program can normally only be tested by a CALL from **SC_Assembler** BASIC if the ORG and PUT addresses used were the same, or if they were omitted so that both defaulted to 32768. Code can be assembled to any address between 32768 and 65535, so the problem only arises for code which will normally run from an address below 32768.

A facility is provided in **SC_Assembler** BASIC for transferring an object code block to its ORG address for testing, but, because the Assembler code resides in the lower addresses, the lowest ORG address which may be used in this transfer is 29000.

It is important that source code and object code files should be saved to disc before testing. If the program has a fault and crashes, the computer may reset itself or have to be reset because there is no response to the keyboard, and the files in memory would be lost.

After assembly, make a note of the ORG and PUT addresses and file length displayed after pass 1, and exit to BASIC.

512K
Line 200 is a DATA line which holds
200 PUT address, ORG address (29000 or higher), file length.

Change the line to the appropriate address o and file length. The letter m entered as a direct command will transfer the block, which may then be tested by a CALL to the ORG address.

256K
Line 99 is a DATA line which holds
99 PUT address, ORG address (29000 or higher), file length.

Change the line to the appropriate addresses and file length. The letter m entered as a direct command will transfer the block, which may then be tested by a CALL to the ORG address.

SC_Speclone
Memory map

Page	Sam addresses	Used for
0	16384-32767	Sam memory
1	32768-49151	Sam memory
2	49152-65535	Sam memory
3	65536-81919	Spectrum 48K ROM 0-16384
4	81920-98303	Spectrum memory 16384-32767 includes Spectrum screen
5	98304-114687	Spectrum memory 32768-49151
6	114688-131071	Spectrum memory 49152-65535

How SC_Speclone works
The Spectrum 48K ROM is modified to scan for the extra Sam keys such as DELETE and the function keys. This code is placed in a free area of Spectum memory between 14446 and 15615. Bytes 11446-14893 are used for the keyscan and other essential code. All Sam keys are scanned. See pages 25-26 for the special uses assigned to some of the keys.

A printer driver routine is proved at Spectrum addresses 14793 to 14826. (34 bytes). This enables the LPRINT command. LliST mimics LPRINT because tokens are not expanded by this routine. The printer driver sends a linefeed after every carriage return. Instructions for disabling the linefeed are on p.26.

An OUT instruction is used to page the Spectrum ROM to Sam address 0 and to use screen MODE 1, the Spectrum compatible mode, at Spectrum address 16384 when switching to Spectrum mode.

The Spectrum NEW command is used as a switch to return to Sam mode, paging out the Spectrum ROM and paging in the Sam ROM and setting screen 1, MODE 4, at the normal Sam screen pages, the last two pages in memory. RANDOMIZE USR 14888 mimics the normal Spectrum NEW while in Spectrum mode.

In Sam mode all Spectrum addresses reside 65536 above their normal Spectrum address, and so a Spectrum address n may be POKEd from Sam BASIC at address n+65536.

In Sam BASIC GO TO 10 returns to the main menu.
GO TO 1 returns to Spectrum mode without resetting Spectrum memory.

APPENDIX 3
FOR THE BEGINNER — EXAMPLES

If you have never before used an assembler, working through the following short examples will help you.

Example 1
A short program to change the border colour to yellow.

1)Put your **SC_Assembler** working disc in drive 1 and press key F9 to load it. On loading, you will be at the Editor screen with the cursor at the top left corner.

2)Type
10border:lda,6 and press Enter.
The line will be reprinted on screen as
00010 border:LD A,6
The program has inserted spaces and placed the various elements of the line in their correct places on screen. You have no need to worry about spacing or using capital letters for the opcodes.

3)Type
20out(254),a Enter
30ret Enter
Your program will appear as

```
00010 border.LD  A,6
00020        OUT (254),
00030        RET
```

4)Type +s Enter
The SAVE SOURCE mode is entered. Bank 1 is shown as holding lines 00010 to 00030, all other banks as 00000 to 00000. We wish to save the whole file, so change the filename to
bordr.src
and press Enter. The code will be saved to disc.

5)Type a Enter
The file will be assembled, and the following information displayed.

```
ORG 32768-32772 (00005)    #8000-#8004    (#0005)
PUT 32768-32772 (00005)    #8000-#8004    (#0006)
LAB 00001
** pass 1 **
```

** pass 2 complete with no errors.

As this is a program which could run from anywhere in memory, we did not enter ORG or PUT addresses and the program used 32768 for both. The program is 5 bytes long and there is 1 label.

6)Type +z Enter
You will enter the Disassembler. If you press Enter, to disassemble from the default address of 32768, you will see the opcodes for our program listed, confirming that it has been properly assembled. Press ESC to stop the listing and return to the Editor.

7)Type b Enter
The program will exit to BASIC.
Enter SAVE "bordr.cod" CODE
32768,5 to save the machine code to
disc.

8)Type CALL 32768 Enter
The border colour will change to yellow, because our program loaded the A register with 6, Sam's number for the colour banana.

9)Press key F4 to re-enter the Editor.

10)Add the following line to the program - you need not enter it in the correct sequence.
5 *list on

11)Type l Enter
The program will be listed to screen, with the new line 00005 in its correct place. If you typed

ll

the program would be listed on the printer

12)Type a Enter
The program will be assembled again, but the assembly listing will be sent to screen. If you changed line 00005 to

*s printer on

the listing would be sent to the printer.

13)Type d 5 30 Enter
The program will be deleted.

Example 2
Using several banks

1)Type in the following lines

```
00100                    LD A,13
00110                    CALL outputa
```

2)Type b2 Enter
The bank number on the indicator will change to 2.

3)Type i 1000 Enter
The line 01000 will appear on screen and every time you press Enter when you complete a line, the next line number will be presented. The blank lines which break up the listing are produced by pressing Enter without typing in any characters.

4)
Type in the following lines

```
01000          outputa:PUSH   BC
01010                 PUSH    AF
01020                 LD      BC,(outvar)
01030
01040           outloop:IN     A,(C)
01050                 RRCA
01060                 JR      C,outloop
01070
01080                 DEC     C
01090                 POP     AF
01100                 OUT     (C),A
01110
01120                 INC      C
01130                 OUT     (C),B
01140
01150                 DEC     B
01160                 OUT     (C),B
01170
01180                 POP     BC
01190                 RET
```

5)Press SS/ESC or SH1FT/ESC to escape from autoline mode

6)Type b3 Enter
Change to bank 3

7)Type i 1000 Enter
Enters autoline mode

8)Type in the following lines

```
01000 ;output a              send A register to printer
01010 ;                      A=0 to 255
01020 ;
01030 ;                      No registers corrupted
01040                 outvar:EQU #5A10
```

9)Press SHIFT/ESC or SS/ESC to exit autoline.

The three banks now each contain specific parts of the program. Bank 1 contains the main program routine, which calls a subroutine. Bank 2 contains the subroutine, which could become a library routine, since it is a procedure which is likely to be needed in many different programs. Bank 3 contains notes in comment lines, which begin with a semicolon, and one line, 01040, which sets up a variable.

Note that the same line numbers appear in banks 2 and a This would have no effect when the program is assembled, because banks are assembled in sequence, 1,2,3, and the line numbers are only significant within each bank If bank 3 is used for explanatory notes, using line numbers which match those for the routine being described can be helpful.

To save only the subroutine and explanatory notes to disc, proceed as follows

1)Type +s Enter

Change the line numbers in the display under the filename box as follows
Bank 1 00000 00000
Bank 2 01000 01190
Bank 3 01000 01030

2) Change the filename to one of your choice

3) Press Enter.
Nothing will be saved from bank 1, because that is the main routine which calls the subroutine. The whole of bank 2 is saved, and from bank 3, only the comment lines describing the routine are saved, and not the variable which would be applicable only to the main program.

SC_SPECLONE EXAMPLE
PCG's DTP PACK conversion

If you have a disc-based or microdrive version of Spectrum DTP PACK, you must first make a tape copy of the "WM" code block. Reset the Spectrum. Enter CLEAR 24733: LOAD *"m";1; "WM" CODE 54174. When the code block has loaded, enter SAVE "WM" CODE 54174,11362 and save the code block to tape. Prepare a newly formatted disc with only the SAMDOS file on it.

Now, using Sam, load the **SC_Speclone** utility and select menu option 2 - RANDOMIZE USR 0 SPECTRUM. In Spectrum mode, enter CLEAR 24733:LOAD "WM" CODE and play the tape to load the code block.

Type in the following lines of Spectrum BASIC.

```
10 LET D=NOT PI:LET S=D:LET L=D-LET X=D:LET A$="          ":RANDOMIZE
USR 63315
20 POKE 23296,1:GO SUB 60:NEW:RANDOMIZE USR X
30 POKE 23296,2:GO SUB 60:NEW:RAND0MIZE USR X
40 POKE 23296,3:GO SUB 60:NEW:RAND0MIZE USR X
50 POKE 23296,4:NEW:RAND0MIZE USR X
60 LET V=INT (S/256):P0KE 23297,S-(256*V):P0KE 23298,V
70 LET V=INT (L/256):POKE 23299,L-(256*V):POKE 23300,V
80 FOR A=1 TO 10:P0KE 23300+A,C0DE A$(A):NEXT A
90 RETURN
100 POKE 23296,5:NEW:RUN
200 POKE 23296,0.NEW:RUN
300 POKE 65532,158:P0KE 65533,96:RUN
```

In line 10, there are 10 spaces in A$. Lines 20 to 50 POKE a code into 23296, to tell Sam BASIC which DOS operation to perform, and they replace the LOAD, SAVE, ERASE and CAT lines of the original Wordmaster BASIC. The subroutine at 60 pokes the file start and length and filename into variables from which Sam BASIC can retrieve them. Line 100 is a line which will return to Sam BASIC, and line 200 to the **SC_Speclone** menu. Line 300 will clear all files from Wordmaster's memory, and return you to the program, providing a quick way of deleting multiple files. The DOS commands are called normally, from the program's options. To use lines 100-300 you must exit from the program to Spectrum BASIC.

Now enter GO TO 2111, put the prepared disc in drive 1, and select option 5 - SAVE SPECTRUM MEMORY. Give the file name "WM" when prompted. When the Spectrum memory has been saved, press ESC to return you to Sam BASIC.

Type in the following lines of Sam BASIC.

```
1000 LET A=PEEK (23296+65536) :ON A:GO TO LOADER:GO TO
SAVER:GO TO ERASER:GO TO CATTER:STOP
1200 DEF PROC GETVARS
1210 LET S=256*PEEK (65536+23298)+PEEK (65536+23297)
1220 LET L=256*PEEK (65536+23300)+PEEK
(65536+23299) 1230 DIM A$(10)
1240 FOR A=l TO 10:L.ET A$(A)=CHR$ PEEK
(65536+23300+A):NEXT A 1250 END PROC
1500 LABEL LOADER
1510 GETVARS:LOAD A$ CODE S+65536,L:GO TO 1
1600 LABEL SAVER
1610 GETVARS:SAVE A$ CODE S+65536,L:GO TO 1
1700 LABEL ERASER
1710 GETVARS:ERASE A$:GO TO 1
1800 LABEL CATTER
1810 CLS:DIR 1:PAUSE 0:GO TO 1
8000 OPEN#5; "b": PRINT #5;CHR$ 27; "C";CHR$ 70;:CLOSE #5:RETURN
```

Lines 1000 to 1810 perform the DOS operations. Line 1000 PEEKs the variable to discover which operation is required and directs the program to the correct subroutine. Each operation ends in GO TO 1, which returns to the Spectrum program at the command after NEW, which switched to Sam mode. The procedure at 1200 sets up the variables. Line 8000 contains any codes you may wish to send to the printer. The ones given set up A4 paper length, but they may be changed to any you wish to use. If you use the line to send printer codes, the printer must be on line when you load the program

Now alter line 9000 to read

```
9000 CLS #:PALETTE#:CLS #:CLEAR 29999:LOAD "rom" CODE:LOAD
"high" CODE:LOAD "low"CODE:LOAD "WM" CODE:DPOKE
(23730+65536),247330: GO SUB 8000:POKE 88832,0:RUN 1.
```

You have added commands to load the Spectrum memory, POKE the Spectrum RAMTOP and call the printer codes subroutine, and changed the RUN address to 1.You can now save the Sam BASIC to your prepared disc.

```
SAVE "AUTOWORD" LINE 9000.
```

Now enter POKE 80324,195 and SAVE "rom" CODE 65536,16384. Finally, you must copy the "high", and "low" code blocks from your **SC_Speclone** disc to your new disc. The program will autoload, and will be exactly like the Spectrum version except that the tape load/save operations are unuseable. PlusD disc files may be used, but if you have the tape version of DTP PACK, you must copy all the extension programs, fonts, etc. to Sam discs.

INDEX

Text, design and layout by CAROL BROOKSBANK
from technical description by S.J. NUTTING
using SPECTRUM DTP PACK by P.C.G.

AMENDMENTS AND CHANGES

The key scanning on SC_ASSEMBLER is now done through the Sam Rom, with the advantage the 8 character buffer, so now matter how fast you type the editor will always keep up with you. The keyscan variables are now at 23561 for REPDEL and 23562 for REPSPD

During scrolling of the screen (Dissasemble, Text and source listing etc) keys P=Pause, O=Unpause

If you have a 256K source file and would like to load it into the 512K Assembler then there is a file on Disc called CONVERTOR this simple converts the 256K Source file into the 512K source file format.

There are 4 source files on Disc they are :-

opcodes256 and opcodes512 load them in depending which
routine256 and routine512 version of SC_ASSEMBLER you have

You may have some Lerm Sam Assembler source files you would like to use on SC_ASSEMBLER, if so follow the below instructions:-

Load in SAM ASSEMBLER Version 2.0 or Version 3.0 (256K Version), If you have any other version give me a ring and I may be able to help you with other versions.
Now load in a Sam Assembler Source file in at the first Bank (Ram pages 3/4),then QUIT to Basic and then load in from my Disc either lerm2 or lerm3 depending on which version of Sam Assembler you are using. Then on Sam Assemblers editor press P and Return if using Version 2 or PR if using Version 3, then type LIST, once all source is listed to the Screen type QUIT to return Back to Basic. The Converting will now take place, if there is any syntax my Assembler would not except then you will be prompted with the error message and you will need to type out the offending bad line note type the line numbers, labels, opcodes in there respective tab column positions. Once conversion is complete you will be prompted to save the source to Disc. To reload the source into my Assembler use the following Addresses to load the source into which bank, you can load multiple converted source into as many banks as you like:-

Bank 1 98304	Bank 4 196608	Bank 7 294912
Bank 2 131072	Bank 5 229376	Bank 8 327680
Bank 3 163840	Bank 6 262144	Bank 9 360448

Note that lerm sauce must be tabbed in it's right fields :-
Column
0to 4 6 to 22 24 upwards
00010 labels Opcodes ;any remarks are not convey
But remarks in a line only can e.g 00010 ;remark