LERM SAM ASSEMBLER.

ASSEMBLER + DISASSEMBLER + SINGLESTEP

CONTENTS

MANUAL FOR SAM ASSEMBLER VER 3.a 256k VER 3.b 512k.

(Includes a DISSASSEMBLER + SINGLESTEP)

INTRODUCTION

The SAM ASSEMBLER has been written to be easy to use and learn. Any users with our Spectrum Z80 Toolkit will be able to load files from that assembler into SAM ASSEMBLER. One PAGE of RAM memory is 32k bytes long. The ASSEMBLER has two PAGES of RAM available for source files in the 256k version and 8 PAGES in the 512k version plus one page of RAM available for the machine code that the Assembler creates - called OBJECT CODE.

So the SOURCE FILE is the text that you type into the ASSEMBLER, and the OBJECT CODE is the machine code that the ASSEMBLER creates having "read" the source file. Incidentally, we shall use SA as short for SAM ASSEMBLER.

This means that you can have up to 64k or 256k of source file in memory at once depending upon whether you have the 256k or 512k version. Obviously you can put either version into the 512k SAM, but only the 256k version into the 256k SAM. The source files are treated as separate files so if using two files to assemble one object code file then the files have to be joined together using EQU statements (see later). A novel feature is the HELP PAGE which also doubles as a SCRATCH PAD (this is an area that you can write notes into just like a scrap of paper). You can use the HELP PAGE as a calculator to convert numbers from hexadecimal to binary or decimal in any order. Also included in the program is a NUMERIC DUMP and an ASCII DUMP. A feature of this program allows you to convert old 32 column listings to the 64 column used here. Labels can be any length up to 14 bytes and can include the underline character. There is 15k allowed for labels created by your SOURCE FILES, so that you can have as many as 976 labels. (In practice with a full source file of 32k you could have a label every 3 lines). We hope you enjoy using the SAM ASSEMBLER - we believe that it is a first rate product.

GUARANTEE AND FAULTY TAPES/DISKS.

If the tape/disk we supply you with turns out to be faulty then please return it to us for replacement. Please mark your envelope as FAULTY, and return the TAPE/DISK only. If you are not satisfied with the product then please return it, and the manual to us STRAIGHT AWAY, and mark your envelope REFUND (mail order DIRECT FROM US ONLY). The product must be in good condition. An SAE does speed up the process.

CONTENTS OF YOUR LERM TAPE/DISK

The 3 files are the 256k version, and the last 3 are for the 512k version. The order of the files is as follows:

	auto ass3a	- BASIC for 256k
SIDE 1 IF	maincode3a	- machine code for Assembler
ON TAPE.	stepcode3a	- Single step code
	auto ass3b	- BASIC for the 512k version.
SIDE 2 IF	maincode3b	- machine code for 512k assembler
ON TAPE.	stepcode3b	- Single step code for 512k

MAKING A BACKUP TO DISK

First prepare a blank disk using your SAM System disk and make sure that you reply Yes when asked if you want the SAMDOS saved as well. SAMDOS should be the first file on the disk. Now type DEVICE T, insert your LERM tape into your tape recorder, press function key F7 and play on your recorder. Once the program has loaded you will be presented with a menu of options. Press Q to take you back into SAM basic. Now with your prepared disk in drive one, press function key F0. This will save the 256k SAM assembler and single step to your disk. Your LERM tape should be put away somewhere safe now as this is your master and should be looked after. If you have a 512k SAM, then repeat the above with a fresh disk, but only load in the last 3 files.

MAKING A BACKUP TO TAPE

Switch on your SAM, rewind your LERM tape, and press PLAY on your recorder and the F7 key on your SAM. Having loaded in, press Q to return to SAM BASIC. To make a copy to TAPE simply start recording and press the F1 key. Again, for the 512k version, repeat the above, but load in the last 3 files.

MEMORY ALLOCATION

You SHOULD NOT attempt to assemble any code below 32768 as the Assembler uses this memory as its own. Should you require your code to run from an address below 32768 then use the following instructions: ORG 16384 : PUT 32768. This tells the assembler to assemble the code to run from 16384 but to PUT the code it creates into another position, in this case into address 32768. After assembly you would save the code from 32768 and then re-load it to 16384 to test it. (previous versions of SA used DISP rather than PUT but old s.files will automatically be changed to PUT).

When the program is loaded the main assembler code loads to address 65536 and is switched into address zero by a few bytes of machine code at address 65527. These few bytes of code can be overwritten by your code with no ill effect as the basic pokes them into memory before the assembler is entered.

DEFINITIONS

SOURCE FILE.	All typed text including opcodes, labels and any of your own comments.
OBJECT CODE.	This is the machine code produced by the assembler
LABEL.	A pointer to an entry point in the source code.
DIRECTIVE.	These are special instructions to the assembler and only tell the assembler to do some particular job. They will not be assembled as machine code.
SYMBOLS.	A LABEL that has had a numeric value given to it at assembly time - these numbers are stored in a table called the SYMBOL TABLE. e.g. if your source file has 7 LABELS, then the values of these labels and the label name is stored in the SYMBOL TABLE.

(an OPCODE is short for OPERATION CODE - examples of these are LD A, or LDIR, etc.)

ASSEMBLER INSTRUCTIONS.

All opcodes and instructions must be in upper case. You can have multi-statement lines as in basic separated by a (:) character. You can have comments on a line separated from the instruction by a (;) or by a (*) character. LABELS can be any combination of upper case lower case or numerals but must start with an alphabetic character. You can also use the underline character but not a space. You can have your listings laid out in any way that you like for instance the below is an example of one instruction laid out in different ways all are valid and will assemble ok.

```
00010 START LD A,10
00020 START         LD A  ,10
00030 START         LD    A,10
00040 START         LD A,           10
00050 START                                LD A,10
```

The below are all DIRECTIVES as they tell the assembler what to do, and are not part of the assembled code.

ORG address — This is the instruction you use to tell the assembler where your code should RUN from when it assembles. IMPORTANT- YOU MUST NOT USE ORG BELOW 32768 unless you use a PUT instruction (see below).

PUT address — This tells the assembler to PUT the code at the PUT address but to assemble it for the ORG address. For example, you may wish to assemble your machine code to RUN from address 28000. You would do the following:
10 ORG 28000
20 PUT 32768 ; Or any address between 32768-65530
You could also have done:
20 ORG 28000:PUT 32768
IMPORTANT - YOU MUST NOT in any circumstances use a PUT value BELOW 32768.

EQU — Means EQUATE or equals. A label may have a value assigned to it. Useful if you want to refer to a point outside of the source code.
This takes the form:
PETER EQU 55000 or NEW EQU 00000 or FRED EQU 2
JP NEW would mean jump to address zero.

DB n,n,etc — DEFINE BYTES. Inserts 8 bit data at the current assembly address separated by a (,).
eg: DB 1,2,3,4,255 etc.

DW nn,nn etc — DEFINE WORDS. Inserts 16 bit data (addresses) at the current assembly address separated by a (,).
eg DW 12345,32768,49152 etc.

DM "message" — DEFINE MESSAGE. Insert the string at the current assembly address enclosed by quotes (").
eg DM "This is a message"

DS nn — DEFINE SPACES. Reserves nn spaces at the current assembly address.

eg DS 50 will skip 50 bytes before continuing to assemble the next instruction.

$ Dollar symbol. This will tell the assembler to use the current assembly address.

```
eg   LD B,20
     DJNZ $
```

The dollar has been used instead of a label and means loop back to the DJNZ instruction.
It could have been written as:

```
00100          LD B,20
00110 LOOP     DJNZ LOOP
```

nn OR & nn Tells the assembler that the number following is a hexadecimal number and must take the form: 0000 to #FFFF.

% 11111111 The percent sign tells the assembler that the following string is a binary number and must take the form %11111111. Leading zeros in the binary number are not required but do make the number more readable by us mere humans. Only eight bit binary numbers are allowed.

NOTE: Spectrum users with our Z80 toolkit program can load the files created on their assembler and any DEFB or DEFW etc will be automatically converted to DB DW etc. Also DISP will be converted to PUT. There is a keyword called TAB this will re-format the listing to allow for 14 byte labels. The TAB command will not re-format lines with DM DW or DB on them nor will it re-format any line with a label on it. There must be 3 spaces between the line number and the opcode for this option to work. TAB can take a parameter between 8 or 21 and will default to column 21.

USING THE ASSEMBLER.

To learn how to use the SAM assembler we will use an example. Once the program has been loaded press (A) to enter the assembler. You will be presented with the LERM logo on the top line of the screen, on the next line you will see various status messages as below:

```
FILE 1  COL 1   F.LEN 2        FREE 32731  CAPS ON   PRT OFF
```

The first message tells you that you are using Source file 1. If you press F0 you will see the FILE message change to 2 indicating that you are now using file 2. Press F0 in the 256k version once again and you will be taken back to file 1. In the 512k version F1 increments the file and F0 decrements the file allowing you to go through all 8 source files. The second message tells you what column the cursor is currently on. Next you are told the length of the file - on start up the file length will be 2 bytes. These 2 bytes are end of file markers. Free tells you how much source file memory you have left. The CAPS ON tells you that upper case characters will be typed in. Pressing CAPS will give the message CAPS OFF so that you can type in lower case. All opcodes must be typed in upper case, so if you have selected lower case for any reason, then when you press RETURN to enter the line into the source file, the program

will automatically toggle back to upper case input. The last message tells you whether the printer is ON or OFF.

LINE NUMBERS.....AUTO

Like SAM basic, line numbers are needed. This can be done manually by typing the line number then the instruction. The program can produce them for you - type AUTO and press RETURN and a 00010 will appear on the screen. If you now press RETURN again the cursor will drop to the next line and a 00020 will appear, this is your auto line number mode. So AUTO is a DIRECTIVE and tells the assembler to AUTOmatically insert line numbers starting at ten, in steps of ten. You can do AUTO 10 1 and the assembler will produce line numbers in steps of 1. Doing AUTO 134 2 would put in lines starting at 134 in steps of 2. If you have lines in steps of 10 but want add some lines then you can use the following technique.

```
ORIGINAL 130 LD HL,FRED
         140 LD A,10:PUSH AF
```

To add some lines (max of 9) between these use AUTO 131 1 and press ENTER. You will see that SA creates lines 131, then 132, etc, for you automatically. Don't go beyond 139 or you will overwrite your original line 140! You can then RENUMBER the lines (see later) Note that earlier versions of SA used INSERT which we have changed to AUTO.

CLEAR SCREEN.....INV key or CLS

Press the INV key, and the screen will clear. If you now type AUTO and press RETURN you will once again get a line 00010 on the screen. Alternatively enter CLS and press the RETURN key.

TABULATE.....TAB key

Pressing the TAB key will take the cursor to the next TAB position in the line. This has been programmed to make the second tab position col 21. This allows 16 spaces between the line number and the opcode so that labels up to 14 bytes can be accommodated.

BRACKETS ( ) and underline...F4 F5 and F6

For convenience the function keys F4 and F5 will print open and closed brackets. The F6 key will produce the underline character. This saves you pressing the shift keys to access these often used characters.

With line 00010 on the screen type in the listing in FIG.1 not forgetting to use the tab key. You must press RETURN after every line otherwise the line will not be inserted into the file.

NOTE: When the cursor reaches col 64 it will stop moving. You should press RETURN to enter the line. Any character at col 64 will be rejected. Only 63 characters are allowed on a line.

Once the listing has been typed in you can move the cursor to any position on the screen with the CURSOR KEYS and make alterations, but the RETURN key must be pressed for the line to be entered into the source file. It will not be entered if you just make an alteration and simply use the cursor keys to move out of the line.

FIG.1
```
00010 ORG 16384 : PUT 32768
00020 HMPR          EQU 251 ; High Memory Page Register
00030 VMPR          EQU 252 ; Video Memory Page Register
00040 START         LD HL,32768
00050               LD DE,DATA
00060               LD B 0
00070               IN A,(HMPR)
00080               PUSH AF
00090               IN A,(VMPR)
00100               AND 31
00110               OUT (HMPR),A
00120 LOOP_1        LD A,(DE)
00130               LD (HL),A
00140               INC HL
00150               DJNZ LOOP_1
00160               POP AF
00170               OUT (HMPR),A
00180               RET
00190 ;
00200 DATA          DB 255
00210 END           EQU $
00220 LENGTH        EQU END-START
00230 _
```

CLEARLINE.....EDIT key

At this point the cursor will be flashing on line 00230. If you now press the EDIT key the line containing the cursor will disappear and so CLEAR the line. Press RETURN now and you will be back in command mode. If you now move the cursor into any line on the screen and press EDIT the line will disappear. Now press the INV key (cls) and type LIST. So the EDIT key clears the current line but the line that was already present in that position is NOT deleted from the source file.

LIST.....LIST

Typing LIST, then pressing RETURN, will list from the FIRST line of the source file. Once the screen has filled up pressing RETURN again will LIST the next screen. Pressing EDIT will exit from the LIST mode. Typing LIST 140 will list the source from line 140 onwards. Listing can also be done from a label. e.g Enter the command LIST START, then press the RETURN key. You will see that the file has listed FROM the Label START. Having typed list you will see that the line you CLEARED using the EDIT key is still in the source file. So EDIT is the clearline key and will clear the line but not remove it from the file.

Note that you can only enter LIST followed by a label name provided that this label is placed in the FIRST TAB position.
e.g if you have

```
00100 FRED       LD A,2
00105      PETE  LD HL,65000
00110 JOHN       LD (GAP),HL
```

You can use LIST FRED, or LIST JOHN, but not LIST PETE.

If you are looking at a list AND THE CURSOR IS ON THE BOTTOM LINE OF THE SCREEN, then by pressing the CURSOR DOWN key, the next line of the listing will appear on the bottom line, and the screen will be scrolled up. Similarly, if the cursor is ON THE TOP OF THE SCREEN and you press CURSOR UP, the list is moved

down and the previous line will appear on the top of the screen.

If, at the start of a line you enter EDIT then a valid line number like so: EDIT 400 then return
line 400 will appear on the bottom of the screen. If you then pressed CURSOR DOWN, the next line number will appear on the bottom of the screen, and so on. If you place the cursor on a line number then press the F3 key, again a listing of that line number will appear at the bottom of the screen, and use of CURSOR DOWN will make the next line number appear and so on.

If you go to the HELP PAGE by pressing F8, and type in a LINE NUMBER, then move back the cursor to the start of the line number and press F3 you can see a listing of your source file within the HELP PAGE. Pressing F8 toggles from the assembler to the help page, and back again. The colour of the PAPER in the assembler if black, but in the help page it is blue. If you write over a listing IN THE HELP PAGE it does NOT change the actual source file, so that when you return to the assembler from the HELP PAGE it will remain as it was. You can also list from a label within the help page. This can be very useful for re-veiwing another source file or a routine in the file you are working on.

ASSEMBLING.....ASSEM

At this point we will Assemble the SOURCE FILE INTO MACHINE CODE. To do this type ASSEM, the screen will clear and the message Assembling file 1 will appear. If you have typed in the listing from Fig.1 correctly including the mistake at line 00060 then you will see the message : Error 4: Comma or bracket expected. A space will be left and then the offending line will be re-printed for you to rectify. The line will be re-printed as follows: 00060 LD B 0. A comma is missing between the B and the zero, so use your cursors to move into the line and placing the cursor between the B and the zero type the comma in. Now press RETURN and then LIST. The comma will have been inserted into the file so we can once again attempt to assemble the file. Type ASSEM again and if all is ok you should see the message "Assembled ok". This tells you that the source file has been assembled and the code will be at address 32768 waiting for you to save it. Before we save the source file and the object code we will look at some other functions of the assembler. SA has been set with a default setting of 5 errors, so after entering ASSEM, if there are any errors in your source text, the first 5 will appear on the screen. Pressing RETURN will give the next 5 and so on. You can enter ASSEM 2, and any errors will be printed two at a time.

RENUMBERING.....RENUM

Type in the following line:

00025 * This is a comment line

Use the CAPS key to enter the lower case characters and when you have finished typing press RETURN. If you now type LIST you will see that line 00025 has been inserted into the file at the correct place in the listing. Now type RENUM (RETURN) and then LIST again. You will see that the file has been re-numbered from line 10 in steps of ten and that line 00025 has now become line

00030. You can change the step size of 10 - see appendix.

INSERT/DELETE.....F9/F7
Move the cursor into line 00030 and between the (*) and the (T) and press function key F9. This will insert a space at the cursor position. If you continue to press the F9 key, the line will eventually be pushed off the end of the line. Before you do push the line off the screen, stop, and press function key F7 - this will pull the line back towards the cursor and the characters will disappear underneath the cursor. In effect these two keys are F9 insert character at the cursor position and F7 delete a character at the cursor position.

SYMBOL TABLE.....SYM
At this point we will go back to our source file and the object code. We want to save both the source file and the object code but before exiting to the menu we will need to know how long the object code is. To find out type SYM. This will print the symbol table for you. SYM tell you the values that SA has calculated for you can and gave to the LABELS you have created in your s.file. At assembly time all the labels had addresses or numbers assigned to them - you should get the following list:

LENGTH	001B	27
END	401B	16411
LOOP_1	4011	16401
DATA	401A	16410
START	4000	16384
VMPR	00FC	252
HMPR	00FB	251

As you can see the label LENGTH tells us that the length of the object code is 27 bytes long. Have another look at the listing to see how this works out. Also note that label START is at address 16384. This was our ORG address but we used a PUT 32768 so in fact our object code is at address 32768 waiting to be saved. It will need to be loaded back to address 16384 before we can CALL it to see if it works.

VIEW ALL LABELS.....LABEL
You can look at all of the labels in the file by typing LABEL. The screen will clear and the first page of labels will appear in the order that they are in the file. Typing LABEL A (RETURN) will list all labels starting with the character A. Pressing RETURN will list the next page of labels. Pressing ESC will abort this option and you will be told how many labels there are up to that point. If you continue to press RETURN until all the labels have been listed then you will be told how many labels the file contains.

LAST LINE OF SOURCE.....LAST
Typing LAST will tell you the last line in your source file. Useful if you wish to continue writing source from the bottom of the file.

EXIT TO MENU.....QUIT
Typing QUIT and RETURN will take you back to the start up menu.

PEEK into memory.....PEEK, DPEEK
If you want to look at the contents of any of the addresses from 0 to 65535, simply use the command PEEK or DPEEK, then RETURN.
e.g. Type PEEK 50000 (and RETURN) or DPEEK 53000 (and RETURN).

TAPE USERS.....
Tape users should read through the following paragraphs but note that when an option to load or save is selected then the directory will not appear on the screen.

NOTE: THE OPTIONS TO LOAD, SAVE, AND ERASE, TO DISK, EITHER SOURCE FILES OR OBJECT CODE HAVE 2 POSSIBLE KEY PRESSES. IF YOU PRESS THE LOWER CASE, THEN A "DIR" OF THE DISK IS TAKEN BEFORE YOU ARE ASKED FOR A FILE NAME. IF YOU ENTER THE UPPER CASE, THEN NO DIR IS DONE, AND YOU ARE ASKED TO ENTER THE FILE NAME STRAIGHT WAY. e.g. lower case "s" does a DIR before asking for the name to save the source file. Pressing capital "S" avoids the DIR.

SAVING SOURCE FILES.....S
There are various options from this menu but for now we want first to save our source file, so press (S). The screen will clear and you will see another menu on the screen. There are 4 options available to you in the 256k version.

1 save the source file in file 1
2 Save the source file in file 2
D Save a double file
Q to quit back to the main menu.

As our source file is in file 1 then we will select option 1. You will first be presented with the directory. Then you will be asked for the file name. The file name should not be more than 7 characters long. Options 1 and 2 will save files with a (.SF) tagged onto the end of the name, and when the directory is called up from option 1 or 2, only the files with .SF will be shown. At this point pressing RETURN will take you back to the main menu. Once the file name is entered and the return key is pressed, the file will be saved and verified. A return to the MAIN MENU then takes place. If there is not a file available for saving when you select your choice, then you will be suitably informed. Saving from file 2 is the same as for file 1.

If you have 2 files in memory you can save them as separate files or go for the double file option. If you use the double option you are effectively saving the two separate files as one single file. This option will save file 1 as 32768 bytes long and then tag file 2 on to the end, so saving both files as a complete block. A double file will be saved with a (.DF) on the end of the name and on selecting this option for saving or loading then the directory will only show files with .DF on them. You cannot load a double file as a single file. For saving menu of the 512k version - see later.

MERGE SOURCE FILES.
Two files can be merged together by using the merge option from the main menu. This will load in your selected file and join it

onto the end of your existing FILE 1. You must be sure that both files when joined together do not exceed 32730 bytes in total. Having merged the file onto the end of FILE 1, your first task must be to go into the Assembler then type RENUM and RETURN. FILE 1 will be renumbered and you can now list the combined file and work on it.

LOADING SOURCE FILES.....L
For the 256k version loading source files is the same as for saving them. There is no need to specify the .SF or .DF - simply enter the file name to be loaded excluding the .SF or .DF

The directory will only show files that can be loaded. i.e. if a double file has been selected, only files with (.DF) will be shown. When a source file is loaded the assembler will be automatically entered. The first page of the file will be listed. Pressing return will continue the listing .or pressing the EDIT key will abort listing. If you load a file into file 2 then you will have to select file 2 from the keyboard before the file will list.

For LOADING in the 512k version - see later.

SAVING OBJECT CODE .....O
When this option is selected the full directory is called up and you are then asked for the name of the file. This can be up to 10 characters as normal. You will then be asked for the START address of the code - this will be above 32768. You will now be asked for the length of the code. This is the value you took from the symbol table and in our example program is 27 bytes. Once the code has been saved it will be verified and you will be returned to the main menu. Note that if you used PUT, that is the address from which you must START saving.

EXIT TO BASIC.....Q
Pressing (Q) will exit to Sam BASIC. We will try our machine code now. Type LOAD "filename" CODE 16384. Your object code was ORGed for address 16384 so this is where we will load it to. Once loaded type CALL 16384:PAUSE (RETURN). You should see the top two lines of the screen fill with long white lines. If you now press RETURN the ok message will appear. What this piece of code has done is to page your screen into address 32768 and poke 255 into 256 locations starting at the top of the screen. It has then paged back the area of RAM that was previously at 32768 and returned to BASIC.

RE-ENTERING ASSEMBLER.....F4
Pressing F4 will run the program and you will be back to the main menu.

DIRECTORY.....D
Pressing (D) will call up the full directory and then wait for you to press a key and return to the main menu.

ERASING FILES.....E
Pressing (E) will allow you to erase a file from the disk. You must type in the full name of the file to ERASE. e.g.If you wish to erase a source file with the (.SF) on the end of the file name then you will have to type the filename and the .SF.

SELECT DEVICE T or D.....M

Pressing (M) will allow you to select tape or disk as your saving/loading device. When the tape option has been selected the directory function will obviously not work.

STARTING A NEW SOURCE FILE. NEW and OLD

Typing NEW from the assembler will (if a source file is already present in the machine), allow you to type AUTO and start writing a new file. The status line at the top of the page will show a file length of zero. If, at this point you wish to retrieve your OLD source file and before you have started typing a new file, then type OLD. Your old file will be retrieved for you. Once you have started a new file then the old file will no longer be retrievable as new end markers will be put into the source file.

PRINTER ON-OFF.....PR

The printer can be toggled on or off by typing (PR) all screen output will then go to the printer as well as the screen. If the printer is not available or not connected then nothing will happen. The printing can be aborted at any time by pressing the ESC key. The printer code sends character 10 to the printer after every carriage return character, so if you find that you are getting a blank line between every line you will need to try our option to remove printing CARRIAGE RETURNS - see later - OR adjust your printers dip switches. SEE YOUR PRINTER MANUAL.

SEARCHING FOR A STRING....FIND

It is useful to be able to find occurences of a string or a label within your source file. To do this TYPE (FIND "string"). For example you may wish to find every occurence of the label FRED. Typing FIND "FRED" will print all occurences of the string on the screen. When the screen is full the program will wait for you to press RETURN, and will then continue the search. If you wish to stop searching when the screen is full just press the EDIT key. All lines on the screen can, if you wish, be edited at this point. FIND will only find the actual string, it will not give the address of a label. An option in the help page will do this for you, but only after assembly.

REMOVING LINE/LINES.....DELETE

Sometimes a block of lines need to be removed. A single line can be removed by typing the line number and then RETURN. A block of lines can be removed by typing the first line number to be removed followed by the last line in the block and pressing RETURN. e.g. to remove lines 20 to 100 inclusive you would type DELETE 20 100 and then press RETURN. The block of lines will now be removed. Typing DELETE with no parameters or DELETE with just the start line will have no effect.

MULTI-STATEMENT LINES.....

Multi-statement lines are allowed as in BASIC by using the (:) character to separate the instructions.

e.g 00010 LD A,2:LD B,A:LD (HL),A etc

This saves source file memory but is not very readable. A (:) character cannot be used after a (;) or a (*) has been used on a line to write a comment.

LABELS.....

Labels can be up to 14 bytes long and can be any combination of upper case, lower case, or numerals - they must not include spaces but can use the underline character. Also they must not start with a numeral (0 to 9). For example START, start, STARt, STArt, STart, Start, StArT, sTaRt are all valid labels and could be used in the same file although it would be a bit confusing. Labels can have an OFFSET (or number) added to them. e.g. JP LOOP+3 or JP LOOP-3. Listing to a label is alowed. There must be ONE SPACE between the keyword LIST and the label name.

HEX DECIMAL BINARY NUMBERS.

You can use hex or binary numbers in the source file. For instance LD A,255 could be written in hex as LD A,#FF or as &FF or in binary as LD A,%11111111. Binary numbers can only be used in their 8 bit form. 16 bit binary numbers are not allowed. Hex numbers can be in 16 bit form. e.g. LD HL,&FFFF. Note that Hex values are preceded by a hash character or a & character, and that binary values are preceded by a percent character. These characters must be tight against the value. i.e no spaces in between them. Line numbers will always be in decimal.

CONVERTING TO DIFFERENT BASES

From within the assembler, if you want to convert a number from one base into another simply type BASE then the number. e.g. to convert 45 into BINARY and hex type BASE 45 (then press RETURN).

THE HELP PAGE/SCRATCH PAD

A novel feature of the program is the HELP PAGE. There are various options from the HELP PAGE including an ASCII dump, NUMERIC dump, memory peek, etc. The help page can also be used as a one page note (or scratch) pad.

ENTRY TO THE HELP PAGE...F8

Pressing F8 will enter the help page. This page uses a different page of memory for the screen so that, on exit back to the assembler, your assembly screen is intact including the cursor position. The status line is replaced on entry with the help page message. You can type messages onto the screen if you wish, possibly a label name that you wish to remember and then exit the help page using F8. Your message will still be visible on next entry to the help page. We suggest that you experiment with this option - it will make it much clearer to you. One useful feature of the help page is the ability to list a source file into the help page. You can only list to a label and only then will the listing appear. The help page is a different colour from the main assembly listing so as not to confuse you about where in the program that you are. When source is listed in the help page you cannot edit it or change it in any way. It is simply a convenient way to review a portion of your source file.

EXIT THE HELP PAGE.....F8

Pressing F8 will exit the HELP PAGE back to the assembler. Your assembler screen will be restored exactly as you left it.

THE HELP PAGE AS A SCRATCH PAD.

The HELP PAGE uses a separate screen in a 256k machine and two separate screens on a 512k machine. To type in a message just precede the line with a (*) character and write the message or reminder. There is no need to press return as the line is not stored anywhere - it is always on the screen when you next re-enter the HELP PAGE. You can re-view the help page from basic by pressing (R) from the main menu. Anything stored on the help page will be displayed. Just press any key on the keyboard to exit back to the main menu. There are 6 options from the HELP PAGE that need to use the whole screen. On a 512k machine these options will use a different page of memory from the actual help page, so ensuring that anything typed into the HELP PAGE will not be wiped out. On a 256k machine the use of these 6 options will clear the help page. The 6 options are ASCII dump, Numeric dump, Peek memory, Single step, Disassembler, and clear object code file.

FUNCTIONS FROM THE HELP PAGE.

HEX, DEC, BINARY CONVERSIONS.

All printed reports on the screen will remain as part of the HELP PAGE. Typing (A) + a number will give you the hex, decimal and binary equivalents of the number. e.g.

```
      A 255 (RETURN) will print
   *  Hex 00FF   Dec 255   Bin High 00000000 Low 11111111
```

Typing A #FF will produce exactly the same line.
Typing A%11111111 will again produce the same report.
Binary numbers can only be typed in as eight bit numbers and the leading zeros need not be typed in.

LOW BYTE HIGH BYTE CONVERSIONS.

Typing (B) + a number will produce the following for example :

```
            B65535   *  65535 = Low 255  High 255
```

Typing B#FFFF will produce the same number.

CLEARING THE OBJECT CODE FILE.

Pressing (C) RETURN will clear the screen and ask if you want to clear the object code file. This is the file where code is assembled to. Pressing (N) will exit back to the HELP PAGE. Pressing (Y) will clear memory from 32768 to 65500 and then exit back to the HELP PAGE.

ASCII DUMP.....D+address.

Typing (D) + an address from 0 to 65535 will produce an ASCII dump for you. This consists of address + any character in the range 32 to 128. If you precede the address with the hash character the address will be printed in hex.

NUMERIC DUMP.....N+address.

Typing (N) + an address will produce a numeric dump for you. If you precede the address with the hash character the dump will be all in hex.

PEEK MEMORY.....T+address.

Typing (T) + an address will produce the equivalent of the basic lines 10 LET A=address

```
      20 PRINT A;TAB 6;PEEK A;TAB 11;CHR$ PEEK A
      30 LET A=A+1 : GOTO 20
```

Again preceding the address with a hash + a hex number will print out in hexadecimal.

DISASSEMBLER.....Z+address.

SINGLESTEP.....X+address.

All 6 above options will fill the screen and wait for the RETURN key to be pressed or the Q key to exit. Also, if using a 512k machine, they will use a separate screen from the help page.

SOURCE FILE STATUS.....S.

Pressing (S) + RETURN will produce a source file status report of the currently selected source file as follows:

```
* STATUS FOR FILE 1
* Start of source..... 98304
* Length of source..... 00002
* Free memory..... 32731 Bytes
* Last line of source.. 0
```

If no file is present in the selected page then you are told NO SOURCE FILE PRESENT. In the 512k version of the program the start of the source file is not reported. See summary at the back of this manual for the addresses.

FIND ADDRESS OF LABEL....F"LABEL

Typing F"label (RETURN) will AFTER ASSEMBLY, tell you the address assigned to that label when the assembler created your machine code. This is useful if you wish to find the entry point to a particular routine or require to peek or poke a byte from basic. Do not put quote marks at the end of the label - just at the start. e.g. to find a label called FRED you would type F"FRED (RETURN). If the label does not exist you will be told "Not found". This will only work just after assembly.

PRINTER.....PR.

If the printer option was selected when you entered the help page then all output will go to the printer as well as the screen. The printer can be turned ON or OFF from the HELP PAGE just by typing (PR).

ENTERING THE DISASSEMBLER ..... Z.

While in the HELP PAGE simply press the (Z) key then the address at which you want to start DISSASSEMBLING then press RETURN. You can put a HASH sign before the address for a HEX address. The DISSASSEMBLER can work on any code in the SAMs memory, including itself, but normally you would use it to dissassemble CODE you have loaded into address 32768. So, to dissassemble some machine code, load it into SAMs memory from BASIC something like so:

```
LOAD "fred" CODE 32768
```

then press the F9 key to give you the MAIN MENU. Press (A) to enter the ASSEMBLER, then F8 to enter the HELP PAGE. Now enter

```
Z32768
```

and RETURN. The DISSASSEMBLER will produce a page at a time of dissassembled code. It also shows the undocumented ZILOG codes (see later). Within the DISSASSEMBLER the following operates:

P - Toggles the PRINTER on/off

U - Toggles UPPER/LOWER case - yes you can see the dissassembled code in Upper or lower case!

T - Toggles between giving a DECIMAL or HEX output.
RETURN - Prints out next page of dissassembled code.
Q - Quit dissassembler and goes back to HELP PAGE.
N - Gives a numeric dump from address at top of screen.
D - Gives ASCII dump from address at top of the screen.
C - Gives continuous disassembly until you press any key it then stops at the of bottom of a page - handy for use with a printer.
ESC - Will halt the disassembler until ESC is pressed again.
A - Plus an address allows disassembly from another address.
CRSR DOWN Gives next line of disassembly at the bottom of a page.
CRSR up Will move the disassembly backwards but by one byte at a time. (nudge)
x - Plus an address gives a displaced disassembly.

The memory available for the code for you to examine is from 32768 to 65526. If you have a 512k machine, this option uses as separate screen from the HELP PAGE as does the ASC and numeric dumps, so that any messages you have within the HELP PAGE will remain unchanged when using the dissassembler.

You will notice when displaying the disassembly, the screen shows not only the address, followed by the numbers and the disassembly, but on the right hand side of the screen appears the ASCII characters of the numbers. If the numbers are outside the ASCII range, then a FULL STOP is printed instead.

Displaced disassembly - we have already explained that you must load your CODE into address 32768 to 65535. But suppose that the CODE that you want to examine normally runs in the lower half of memory - between addresses 0 to 32767! With the DISPLACED disassembly option you can achieve this so:

e.g. suppose you want to examine some code that is say 2000 bytes long, but normally loads into address 16384.

1. Load the code into address 32768+16384 (always regard address 32768 when using displacement mode as address zero) so you would load this bit of code to address 49152.

2. Enter the Assembler and use the F8 key to enter the help page Now enter z plus any address just to get into the disassembler.

3. Now type X and at the prompt enter the address 16384. You will see the code that you loaded at 49152 being disassembled correctly as if it had been loaded to address 16384. If you select the Numeric or ASCII options the original address will be displayed i.e. 49152+. This will enable you to exit to BASIC and directly poke the code without having to work out the displacement.

To sum up, to obtain a disassembly of code that resides under address 32768, always regard address 32768 as address zero. The Sam ROM 0 could be loaded at address 32768 and when you press X from the disassembler you would type in address 0. This would then disassemble as if the code were at address zero.

MANUAL FOR SINGLESTEP

Singlestep has been written for the SAM COUPE to enable you to test and de-bug your machine code without crashing. A single step program will allow you to operate each instruction one at a time and as you do so all the registers and flags will be displayed on the screen so that you can see the effect the instruction has on them. Due to the complexity of the Coupe there are some instructions that have been trapped and do not operate. The OUT (NN),A instruction is one of these. If this instruction were to be encountered and operated then the whole machine could get completely confused. The program is ideal for testing small routines before insertion into the main body of the program. It is also very useful for beginners to machine code enabling them to step through some of the simple bitwise instructions to see what happens. A lot can be done with this program so long as you remember its limitations.

PLEASE REMEMBER THIS PROGRAM IS NOT MEAN'T TO ALLOW YOU TO STEP THROUGH LARGE CHUNKS OF CODE AND IF USED AS INTENDED WILL PROVE TO VERY HELPFULL IN DE-BUGGING YOUR ROUTINES. NO REALTIME ROUTINES CAN BE STEPPED THROUGH E.G. LOADING, SAVING, SWITCHING PAGES, ETC. DO NOT ATTEMPT TO SINGLESTEP THE ROM AS THERE IS ONLY THE ASSEMBLER AT ROM ADRESSES AND A CRASH WILL TAKE PLACE. ALL CODE TO BE STEPPED SHOULD BE ABOVE 32768.

ENTERING SINGLESTEP - press X FROM THE HELP PAGE.

On entry you will see all the registers and flags displayed and the program waits for you to enter a start address. This will be the address that you wish to step from. Once the address is entered the instruction at that address will be displayed and the program will wait for you to press the space key before it is obeyed. Once the space key is pressed then the instruction will opperate and the registers affected will be changed to reflect the instruction. Any changes to register values will be shown as will any effect made on the flags. As Call or Push instructions opperate you will see the address pushed onto a 10 position stack. Also you are told how many addresses are on the stack. If your code POPs one to many addresses from the stack then a negative number will be displayed.

There is an auto step mode built in (press C to start, any key to stop), and if the program is about to RET to an address of zero then the auto facility will switch off and the program will wait for action from you. There is also a breakpoint facility. If in breakpoint mode the program will once again stop if it tries to return to address zero. At any time the registers or flags can be changed manually. When in auto step mode pressing any key will halt the program. When in breakpoint mode pressing (P) will stop the breakpoint operation and display the registers up to that point in the program. Pressing (c) will re-start.

A simple disassembler is available from singlestep and will disassemble from the address at the top of the screen. On exit your singlestep screen will be re-displayed as you left it.

NOTES ON USING SINGLESTEP WHEN WRITING A PROGRAM.

When using the assembler to write your code it is sometimes

useful to be able to test a few instructions. This can be done quite easily using source file 2 on the 256k machine or any free page on the 512k version. It is sometimes the case that your code in the main file is not ready to be assembled just yet so using a different page for your test code is useful. Dont forget only the current file will be assembled on invoking the ASSEM command. Do make sure that your ORG for your test code is well out of the way of your main file. If your test code wants to use a ROM routine you will have to set up a dummy routine. For example you might want to do RST 16 to print out the character in the 'A' register, you could set up a buffer (block of spare memory) and send the character code to it instead of printing to the screen, or you could just use a NOP and make the assumption that as long as the character was valid then it would have been printed. This is really the only drawback to the program (not being able to use ROM routines) but we beleive that in other ways the program will be very useful to you. It has been used on many occasions by us at LERM to debug obstinate bits of code and is much easier than trying to do it on paper!

SINGLESTEP KEY COMMAND SUMMARY. (nn = a number)

A + nn	Will allow you to change the address to step from.
B + nn	Will set a break point - the address will be seen on the bottom of the screen. This is really a fast forward routine once you enter the address of your breakpoint the program will go through the singlestep operations without updating the screen until it reaches the breakpoint address where it will stop and update the screen. Pressing (B) will abort the routine.
C	Auto step until a key is pressed or until a zero is requested by a RETurn address.
D	Enters a simple disassembler from the address at the top of the singlestep screen. Press Q to exit back to the singlestep. Will not affect singlesteps registers or flags.
H	Toggles Hex or Decimal display. Will re-display all registers in hex or decimal. Has no effect on singlestep operations.
P	Will stop fast forward when in breakpoint mode and re-display the screen. Press (C) to resume breakpointing.
Q	Exits singlestep back to the help page.
R	Will allow you to change any single or double register value, also lets you change a specific flag value. Single registers are changed by typing A, B, etc. For a 16 bit register such as HL for instance press SHIFT+H and you can then enter the new 16 bit value. To change a flag value select 1 from the prompt and then (F) and you will be prompted for a flag which will just toggle.
T	Will toggle the alternate set of registers into the current set. This is useful if you want to change an alternate register value. Toggle them into the current set (upper set are always current) and use the R option to change them, then toggle them back to where they were. Toggle upper case or lower case opcode display.
SPACE	Singlesteps one instruction at a time.

EXTRAS FOR THE 512K VERSION

This version allows up to 8 s.files to be loaded into memory. These files could be 8 SINGLE files, which are always saved with the chosen name with ".SF" add on the end. You can LOAD/SAVE into any of the 8 pages of memory.

If you are building up a large program that requires say 5 s.files, then you would want to LOAD/SAVE all 5 in ONE COMPLETE BLOCK. S.Files that use 2 PAGES are saved with ".DF" on the end, 3 PAGES long ".3F", 4 PAGES long ".5F", up to ".8F". These files MUST always load/save from the start of the FIRST file, which the BASIC does AUTOMATICALLY for you. This is not like SINGLE FILES which can be loaded into any PAGE of the 8 pages of memory. PLEASE NOTE THAT THESE BLOCK FILES CAN'T ACCESS THE SAME SYMBOL TABLE, SO THAT IF YOU HAVE A SYMBOL CALLED FRED IN FILE 1 WHICH IS A 23000 AND THIS VALUE IS NEEDED INTO FILE.2, THEN YOU MUST PUT FRED EQU 23000 INTO THE SECOND S.FILE.

When you do a LOAD/SAVE which includes a DIR, then only the directory will only show files ending in ".SF" or ".DF" if you are loading/saving single or double files. When loading/saving blocks of size 3 or more then a complete DIR of the disk will appear. This is because the DOS doesn't seem to like to do a DIR of files ending only with ".5F"!

When you SAVE files, you are shown on the bottom half of the screen, the amount of memory present in each PAGE of memory. When you save the first 3 PAGES, the program will save 32k for the first page, plus 32k for the second, plus the amount of memory used in the third. This is automatic.

LOOKING INTO SAM ASSEMBLERS BASIC

COLOURS

If you look into lines 60 + 65 (25 + 40 for 512k version) of the BASIC you will see the PALETTE colours we have chosen for SA. Naturally you can change them and RESAVE using the F0 or F1 keys for disk/tape.

PRINTER

When using a printer, SA uses micro-print, which selects a font type which is small, and means that more text can appear on a sheet of paper. If you look at line 1500, and do GOTO 1500, then you can turn off this micro-print to "normal" size. You will then RESAVE as usual. To turn the micro-print back ON do GOTO 1515 + RESAVE. The lines for the 512k version are 6000 and 6030.

Some printers are set up in such a way, using DIP switches, so that you always get blank line between all printed lines. If so do GOTO 1515, and RESAVE. This can be reversed using GOTO 1650. For the 512k version they are lines 6060 and 6090.

THE 256K VERSION - SINGLE FILES ONLY

The majority of programs you write will only required one s.file. To save you time having to select (1), (2), or (D) to load/save, you can make a version that assumes that you only ever use ONE PAGE of memory. To do this, go into BASIC, and EDIT lines 415 and 950 to remove the REM. Press F0/F1 to save to disk/tape.

APPENDIX

PLEASE READ THROUGH THIS SECTION, AS IT NOT ONLY GIVES A SUMMARY OF THE PREVIOUS PAGES, BUT ALSO CONTAINS EXTRA INFORMATION. EXPERIEMNT WITH THE OPTIONS FOR YOURSELF AND SEE WHAT HAPPENS.

GENERAL STRUCTURE

From BASIC menu press A to get into the ASSEMBLER.
From ASSEMBLER press F8 to toggle in/out of HELP PAGE.
To get to DISASSEMBLER press Z+number from WITHIN THE HELP PAGE.
To get to SINGLESTEP press X from the HELP PAGE.

ERROR MESSAGES

The assembler generates error messages when an error occurs during assembly. The line that the error is on will be printed on the screen and can be edited immediately.

ERROR 0 : Illegal character or incomplete statement.
ERROR 1 : Labels cannot be more than 14 bytes long.
ERROR 2 : You have missed a bracket from the opcode.
ERROR 3 : The Jump Relative or IX+ or IX- is out of range.
ERROR 4 : You have missed a comma or bracket from the op code.
ERROR 5 : There is a context error in your line.
ERROR 6 : The label has been used more than once.
ERROR 7 : You have missed a bracket from the opcode.
ERROR 8 : An illegal opcode has been used.
ERROR 9 : The label has not been defined yet.
ERROR 10: There is no source file to assemble.

KEY SUMMARY

ASSEMBLER.

FUNCTION	KEYWORD	PARAMETERS	EXPLANATION
ASSEMBLE.	ASSEM	x	Assemble and print x errors (default is 5)
AUTO LINE NUMBER.	AUTO	x y	Start auto line numbering from line x with y steps.
BASE CONVERSION.	BASE	xx	xx will be either decimal hex or binary. All the bases will be printed.
CLEAR SCREEN.	CLS or the INV key		Clears the screen.
DELETE BLOCK.	DELETE	x y	Delete from line x to line y inclusive.
DOUBLE PEEK.	DPEEK	nn	Does a double peek of the specified memory location.
EDIT LINE.	EDIT	xx	Pulls the specified line to the bottom of the screen.
SEARCH SOURCE FILE.	FIND	"string"x,y,z	Find "string" between line x and line y and print z occurencies at a time.

LIST ALL LABELS.	LABEL	A,B,C,etc	Will list all labels in the file starting with 'A' one screenful at a time. RETURN lists next screenfull.ESC exits the listing. LABEL on its own defaults to all labels. On exit the number found will be reported.
LIST FILE BY LINE NUMBER.	LIST	x y	List from line x to line y one page at a time.
	LIST	x,y	Lists y lines starting from line x. Handy for non-stop printing.
LIST LABEL.	LIST FRED		Will list from label FRED if FRED exists.
NEW SOURCE FILE.	NEW		Start a new source file.
RETRIEVE OLD FILE.	OLD		Retrieve old source file if there is one in the machine Only after typing NEW and before a new file is started.
SINGLE PEEK.	PEEK	xx	Do a single peek of the address specified.
PRINTER.	PR		Toggle printer on or off.
EXIT TO MAIN MENU.	QUIT		Exits to main menu.
RENUMBER SOURCE FILE.	RENUM	x y	Renumber the file with the first line as x and in steps of y.
PRINT SYMBOL TABLE.	SYM	x	Print the symbol table x lines at a time.
TAB LISTING.	TAB	x	Tab lines across the screen x not less than 8 or greater than 21. Will not work on lines with labels, DM or DB on them. Used mainly to convert files created on SPECTRUM Z80 toolkit. Default = 21.

ASSEMBLER SCREEN EDITOR

CURSORS	**keys**	Left, Right, Up, Down.
EDIT	**key**	Clear the line containing the cursor, or exit from list and auto insert modes.
F7	**function key**	Delete character at cursor position (pull line).

F9	function key	Insert space at cursor position (push line).
F8	function key	Enter/Exit help page.
F4	function key	Print an open bracket.
F5	function key	Print a closed bracket.
F6	function key	Print underline character.
F3	Function key	Placing the cursor at the begining of a line and pressing F1 will bring that line down to the bottom of the screen.
F0	function key ... 256k	Toggle between file 1 or 2.
F0	function key ... 512k	Move forward into files.
F1	function key ... 512k	Move backwards into files.
INV	key	Clears the screen and homes the cursor.
TAB	key	Move cursor to next available tab position.
CAPS	key	Toggle caps lock ON/OFF.
DELETE	key	Delete character to the left of the cursor.
ESC	key	Aborts printing or assembly.
SYMBOL	+ Function keys	Prints numbers 0 to 9.

HELP PAGE SUMMARY

F8 Enter / Exit the help page.
A plus x print the Hex / Decimal / Binary value of x.
A# as above. Hexadecimal input. e.g. #FFFF
A& as above. Hexadecimal input. e.g. &FFFF
A% as above. Binary number input. e.g.%11111111
B plus x prints the low byte and high byte of x.
C Clear the object code file.
D plus address. Prints a ASCII dump. Q to exit.
D plus hex address. Prints a Hexadecimal ASCII dump. Q to exit
F plus label. After assembly will find address of label.
e.g. F"FRED will find address of FRED.
N plus address prints a numeric dump. Q to exit.
N plus hex address will print a hexadecimal numeric dump.
P Toggles printer ON/OFF.
S Prints the status of the current source file.
T plus address. Prints address + peek of address + chr$ peek address. Q to quit.
Z + address enters DISSASSEMBLER. (Q to return to Help page).
X enters the Singlestep mode.

HINTS AND TIPS

Instead of doing CP 0 you can do OR A - this is because OR A uses one byte, whereas CP 0 uses 2 bytes. Instead of LD A,0 which uses 2 bytes, use XOR A uses one byte. In both cases 'A' will be loaded with a zero.

AN 8 BIT PAUSE LOOP.

```
        LD B,255        ; Loop round 255 times doing nothing
LOOP    NOP             ; until B = 0. then RETurn.
        DJNZ LOOP
        RET
```

A 16 BIT PAUSE LOOP.

```
        LD BC,12345     ; Loop round 12345 times doing
LOOP    DEC BC          ; nothing until A=0. Then RETurn.
        LD A,B
        OR C
        JR NZ,LOOP
        RET
```

SWITCHING A PAGE OF MEMORY INTO ADDRESS 32768.

```
        ORG 16384 : DISP 32768
        IN A,(251)              ; HMPR
        PUSH AF                 ; Store current page.
        LD A,10                 ; Use page 10.
        OUT (251),A             ; Switch page 10 into 32768.

        ; REST OF PROGRAM

        POP AF                  ; Retrieve original page.
        OUT (251),A             ; Switch it back in.
        RET                     ; And exit.
```

BASIC will switch pages automatically for you. For instance the code below is written to run from 32768. You could load it elsewhere (only at the START of a page - e.g. 65536+Y*16384, where Y is 1, or 2, or 3, etc) and call it as required. The basic will bring the relevant page down to 32768, call the code and exit back to the BASIC.

```
        ORG 32768
START   LD A,2
        CALL 274        ; Set stream 2 , the screen
        LD HL,MESSAGE   ; Point HL at string to be printed.
        CALL PRINT
        RET             ; Exit back to BASIC
        ;
PRINT   LD A,(HL)
        OR A            ; Compare with 0
        RET Z           ; Return to calling routine if A=0
        PUSH HL         ; Save the pointer on the stack.
        RST 16          ; ROM routine. prints the character in the
        POP HL          ; 'A' register.
        INC HL          ; Increment pointer to next character.
        JR PRINT        ; Loop until 'A' = 0 then exit.
        ;
        MESSAGE DM "Your name." : DB 0
        END EQU $
        LENGTH EQU END-START
```

Assemble this routine and then save it using the O option from the menu. Now reset the SAM and re-boot the disk by typing BOOT1. Load the code to a memory page boundary. e.g. LOAD "CODE" CODE 180224. This is the page boundary that starts off at page 10 of memory. You run it from BASIC with CALL 180224.

You will see your message printed at the top of the screen. This shows that the BASIC system has moved page 10 into address 32768 and called the code from there. Try other PAGE BOUNDARY addresses for yourself the above formuala helps find them), but make sure the machine is reset every time before loading the code and calling it.

READING THE KEYBOARD

```
10 ORG 32768 or anywhere
20 KEYSCAN LD HL,23611     ;  FLAGS   system variable
30 LOOP    LD A,(23560)    ;  KBHEAD  system variable
40         CP "a           ;  check for lower case "a"
50         RET Z           ;  Return to calling routine if match
60         RES 5 (HL)      ;  Reset bit 5 of FLAGS
70         JR LOOP         ;  Continue until match found
```

This routine will keep looping until the "a" key is pressed.

UNDOCUMENTED CODES

There are quite a few instructions in the Z80 processor that Zilog did not document which are reproduced below. The disassembler will handle them all with no trouble. The assembler will not accept them as mnemonics but they can be entered into a source listing by using the following method. As an example one such instruction is

LD A,IX'I

This means load the 'A' register with the IX's I value. This is why the I and the X is shown separately on the screen. The decimal numbers for the above opcode are 221,124. These can be inserted into the assembler as follows:

```
00100 DEFB 221,124 ; this is LD A,IX'I
```

As you can see we have used a comment, so that you can see what the instruction is supposed to be. The above example is the format that the disassembler will display them in. Note that to use IY instead of IX you should substitute DD for FD or 221 for 253, other than that the codes are the same.
ADC A,IX'I would mean ADD A including the carry flag, to the HIGH byte of IX, or ADC A with IX's I.

OTHER LERM PRODUCTS FOR THE SAM

SAMTAPE - our superb SPECTRUM emulator. The "standard" one used by many SAM owners.

SAMDISK - an essential program for all disk owners. Super COPYING routine, repairs disk, special format, fast erase, hide, protect, uneraese, supports 256+512 SAMs and TWO DRIVES, plus much more.

SAM ADDRESS and PHONE manager. Stores names, addressess and phone numbers. Prints labels, fast search, etc. Handy for easy Telephone directory, printing friends names/addresses for X-mas list, keeping track of customers in a business, etc.

FOR MORE INFORMATION, SEND A STAMPED ADDRESSED ENVELOPE TO:
LERM SOFTWARE, 11 BEACONSFIELD CLOSE, WHITLEY BAY,
TYNE AND WAER. NE25 9UW. TEL (091) 2533615.

LERM SA3-v1 - 1290.