**SC_MONITOR v1.2 Fist released April 92, latest ver 1.2 July 1992.
Then came the SC_MONITOR pro pack in March 1994 with
Simon Owens excellent TurboMon.**

This is a program that I did not include with my first Sam program SC_ASSEMBLER in April 1990 as this already included a dissembler as well.

I did write other software between the Assembler and the Monitor program as I was drained with that and felt like doing something different and needed for the Sam.

Very little has changed between versions in the short 3 month time, due to them being bug fixes, my special thanks goes to GlenSoft and there SCADS program, as they were testing there code development with SC_MONITOR and they spotted the bugs with registers and instructions not showing the correct values.

Another special thanks later went to Simon Owen with the inclusions of TurboMon which of course is now embedded within Sim Coupe.

The program was offered without royalties as Simon wanted as many people to have it as possible so it was included with my SC_MONITOR program at no extra charge for customers if I remember right.

I included both programs on the same disc and spent a bit more expenditure on photocopying the manuals and the extra postage to send, but that didn't matter to me as it's getting great software to people who can enjoy and perhaps get upcoming programmers into machine code and develop the Sam software scene.

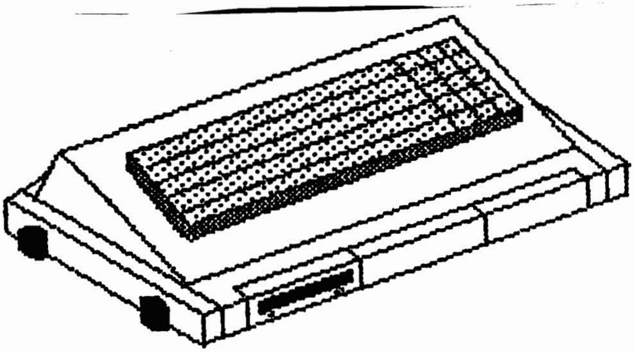
Although my Monitor program had some good features that wasn't in TurboMon, I must admit Simons program was a lot faster and better utility overall.

The manual was produced by my SC_DTP program and outputted by a 24 pin printer.

SJ Nutting *STEVES SOFTWARE April 1990 - January 1996*

SC_MONITOR

Powerful Machine code Utility Software
FOR THE SAM COUPE 256/512K COMPUTER



SC_MONITOR is quite an essential piece of Software, to use with an ASSEMBLER.
Debug & Dissasembles machine code progs
Comprehensive Breakpoints, Traps errors
Works with all opcodes including paging

by S.J.NUTTING
Copyright April 92

STEVES SOFTWARE

7 NARROW CLOSE
HISTON
CAMBRIDGE
CB4 4XX

TEL 0223 235150
From 6pm-9pm

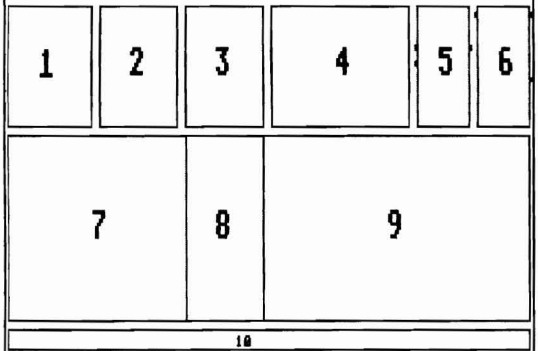
SC_MONITOR GENERAL INFO

This utility will help you debug machine code programs. You can run and keep control over the running of the routine you would like to test, as each opcode executes the registers, stack, dissasemble and memory are updated and displayed on screen, even sam paging will work without crashing program, or you can set up breakpoints to automatically stop at certain addresses or if certain registers hold certain values. There is also tests made to reduce the chance of resets such as if the program jumps to address 0, or the stack is over popped.
There is also a super fast dissasembler which can even go backwards intelligently, it also has ram displacement data and text switchover, plus smooth pixel scroller.

```
 250   95  |01011111 |IX 0     |243 195 50163.. |       |8192
 251   1   |00000001 |IY 0     |243 195 50163.. |       |
 252   30  |HL'0     |HL 40000 |34  210 53794". |50000  |
 76543210  |DE'10    |DE 0     |243 195 50163.. |4      |
 SZ-H-PNC  |BC'0     |BC 0     |243 195 50163.. |       |58
A00101000  |AF'0     |AF 40    |195 230 59075.. |       |
F00000000  | A'0     | A 40    |00101000 4      |       |

1   32768              |PC 32768|32768 243 237 115 42  128 ..s*
                       |SP 20096|32773 195 5   130 62  0   ...>
32768 DI               |        |32778 211 250 49  255 127 ..1.
32769 LD   (32810),A   |   16384|32783 251 205 76  64  243 ..L@
32773 JP   33285       |   500  |32788 62  1   211 251 49  >...
32776 LD   A,0         |   MON  |32793 255 191 195 30  128 ....
32778 OUT  (250),A     |        | MEM   1  42484
32780 LD   SP,32767    |        |58868 115 32  82  69  84  s RE
32783 EI               |        |58873 85  82  78  32  116 URN
32784 CALL 16460       |        |58878 111 32  67  111 110 o Co
32787 DI               |        |58883 116 105 110 117 101 tinu

>q
```



The above two screenshots show the layout of the text, the numbers are there to represent each part of the screen, these will be referred to as zones.

ZONE 10 The long thin box near the bottom of the screen is the main area where commands and inputs are entered.

ZONE 1 Displays the 3 main ports with there values :-

250 is LMPR Low memory page register norm holds 94
251 is HMPR High " " " " " 1
252 is VMPR Video " " " norm holds 30

Also held in this zone are the flag settings for the normal A register and F register.

ZONE 2 holds the binary value held in ports 250 and 251 The values held in the Exchange registers.

ZONE 3 holds the values in the main registers.

ZONE 4 holds the Peek, Dpeek and Ascii values held from the register value in zone 3.
For example look at the screen example to the left, at the top of zone 3 & 4 you should see :- IX 0 243 195 50163.. The 243 byte is the Low value of PEEK IX, The byte 195 is the High value of PEEK IX, The byte 50163 is the Low and High value of DPEEK IX, The two dots represent the Ascii Low and High value of DPEEK IX, as the bytes are higher than 127 they are represented as dots, so are values less than 32, other bytes from 32 to 127 are represented in there Ascii.

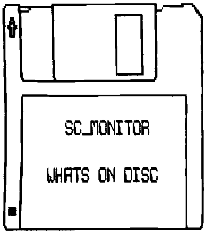
ZONE 5 holds the breakpoint stop values of each register in zone 3. This means that if you are running a machine code program it will stop if the value of the register held in zone 3 and 5 are the same, So if in the example to the left, HL=40000 in zone 3 and the value held in the zone 5 holds 50000, if HL= the value held in zone 5 are the the same, the running of the program will stop.
To complicate things further you might just be able to see two tiny little notches sticking out the sides of zone 5, near the value 50000, this indicates that both the Low and High value of HL must =50000,in another example if we replaced 50000 with say 100 and just one notch is to the left then the Low part of HL must =100, if the notch is to the right then the High part of HL must =100.

ZONE 6 This is similar to zone 5,except the values held here will be compared to the PEEK and DPEEK registers of zone 3.

ZONE 7 This is the Dissasemble zone the byte at the top left hand side is the Ram page no (0-32), after this is the address (0-540671), beneath that are the 9 opcodes from that address.

ZONE 8 shows the value of the Program Counter (PC) this is the start address of the machine code you would like to test.
The Stack Pointer (SP) is the stack address where the last push or Pop was placed.
Beneath this are the values held on the stack, the top most value being the last address on the stack.
If MON appears on the stack, then the next value Popped of the stack by say a RET instruction the test program running will stop running and go back to Monitor mode.

ZONE 9 The fist top 6 text lines are the byte and Ascii text from the address in zone 7.
The last 5 lines are the definable, after MEM is the Ram page number followed by the address from 0-540671, then the normal addresses (32768-49151 plus the bytes and Ascii



1	samdos2	17	C 491520,8192
2	auto	5	BASIC 10
3	moncode	65	C 32768,32768
4	dump	2	C 20224,512
5	softwrite	5	BASIC 10
6	moncode+	65	C 32768,32768
7	DEMO		demo of software

On loading up "auto" you will be able to modify the Monitor coding for linefeeds, palette colours etc.

Then you can save out the modified version to another disc, for your own use.

The Monitor coding is 32K long which can be loaded anywhere as long as it starts at the beginning of a Ram page.

SC_MONITOR requires another screen to work with, it will automatically Open up a screen 2, after the monitor code providing the monitor loaded in an even Ram page number, else it will open up a screen 16K after the monitor code

For example If the Monitor loaded in at Ram Page 2 then-
Ram page 2/3 Area holds the Monitor code
Ram page 4/5 are set up to hold the screen

Another example If the Monitor loaded in at Ram page 1:-
Ram page 1/2 Area holds the monitor code
Ram page 3 is skipped and not used
Ram page 4/5 are set up to hold the screen

The reason for the strange gap is the way the Sam can only work with Screen areas starting with an even number

To Run the Monitor just CALL the address where it loaded. For example if the Monitor loaded in at Ram page 1 thats address 32768, you just CALL 32768.
(also see page 5 on Ram Pages and Address).

When you first enter the Monitor, the Registers will be zeroed out, Port 250=95, Port 251=1, Port 252=30 or14 Zones 5 and 6 are cleared, PC and addresses are set to 32768. SP usually 20100, with MON on the stack.

If however you quit and reentered the monitor, the registers etc are left alone just as you exited form it.

Zone 10 will have an arrow pointing to the right to tell you are in command mode, SC_Monitor is expecting a string of text to be inputted or just a press of a Function key.

The string of text to form a command may need addresses and numbers to complete the command, some commands are just a single letter. You will need to press RETURN to execute the command, Once done the text in zone 10 will stay there until you type a letter at the start of the text, which will cancel any text already there, or you can alter the text with the cursor keys etc, so long as you don't press a letter at the start.

For example type h then press Return, you should see the numbers on the screen toggle from decimal to hex and vice versa, you can just press Return again to alter the screen, if you type in a letter over the h key all text in zone 10 is cleared ready for the next command.

THE COMMANDS

h	toggle hex and Decimal display.
clear	clear out the exchange and normal registers
250=	set Port 250 to hold number after the equals
251=	set Port 251 to hold number after the equals
252=	set Port 252 to hold number after the equals
reg=	reg can be of the exchange or normal registers, as a both part of the register or by single registers,the number after the equals will make that register hold that value e.g HL=40000 normal register HL=40000 E=201 normal register E=201 B'=10 exchange register B'=10 IXl=56 normal register IX low part =56 IYh'=1 exchange register IYh' high part=1
dec	decimal number on its own will display the hex and binary of that decimal numner
#hex	# plus a hex number on its own will display the decimal and binary of that hex number
%bin	% plus a string of 0's or 1's to represent a binary number is converted to hex and dec
reg	a 2 part register (e.g HL IX') on it's own will display the dec,hex and binary of what is held in that register.
pc=	number after the equals is placed into PC
sp=	number after the equals is placed into SP
s0	Res the sign flag in zone 1
s1	Set " " " " " "
z0	Res the zero flag in zone 1
z1	Set " " " " " "
h0	Res the half carry flag in zone 1
h1	Set " " " " " " "
p0	Res the parity/overflow flag in zone 1
p1	Set " " " " " " "
n0	Res the add/subtract flag in zone 1
n1	Set " " " " " " "
c0	Res the carry flag in zone 1
c1	Set " " " " " "
scr	view sams standard screen 1 used by basic or current open screen, if scr is followed by a number, then that screen Ram page number is viewed, Press RETURN to get back to Monitor.
push	if a number follows push then that number will be placed on the stack. If a register is placed after push then the value in that register is placed on stack. registers excepted IX, IY, HL, DE, BC, AF, A
pop	pop on it's own will pop the last value placed on the stack and discard it. If a register is placed after the pop then the value would be placed in that register. registers excepted IX, IY, HL, DE, BC, AF, A if MON appears on the top of the stack then no more pops can be taken off the stack.

BREAKPOINT COMMANDS

sreg= s followed by a normal register as both or single parts with an equals, plus a number after, will set the stop register in zone 5. For example shl=3000 will set the HL stop register with 3000, sc=8 will set the low part of BC stop register to 8, sxh=3 will set the high part of IX stop register to 3.

csreg this is similar to the above but will clear out the stop register and cancel it.
use the 2 parts of the register such as cshl csbc etc but not cse as its single.

spreg=sp followed by a normal register as both or single parts with an equals, plus a number after, will set the stop peek register in zone 6. For example spde=1000 will set the DE stop peek register with 1000, sph=45 will set the high part of HL stop peek reg. syl=12 will set the low part of the IY stop peek register with 12.

cspreg this is similar to the above but will clear out the stop peek register and cancel it.

brk brk on it's own will list the 10 breakpoints in zone 9. If a number from 0-9 follows brk then that breakpoint number will cancel.
If an address between 0-32767 and the Low ram page seperated by a comma OR
If an address between 32768-65535 and the High ram page seperated by a comma follows the breakpoint number, then that breakpoint

will be set from the address and ram page no
E.g brk3 will cancel breakpoint number 3.
brk5,40000,3 will set breakpoint 5 to addr 40000 and High ram page number 3.
brk3,24576,1 will set breakpoint 3 to addr 24576 and Low ram page number 1. he Ram

page may be omitted and the default will be0

o this makesqaitscreenthdumpntootheoprinter
dump of what appears on the screen.

MEMORY LIST MODE

m this will take you into memory list mode the commmand area disappears, certain keys will effect what will happen in the bottom part of zone 9.
The number after the MEM shows the Ram page number, after this is the basic equivelent poke address (0-540671).
The bottom 4 lines show the address,data and Ascii for that particular Ram page number.
The address can be from 0-65535.
Address 0-16383 is the equivelent of viewing the Ram page text in section A of the coupe.
Address 16384-32767 for section B.
Address 32768-49151 for section C.
Address 49152-65535 for section D.

h key will toggle memory list from Decimal to Hex and vise versa.
cursor key up will decrease the address by 5 bytes
cursor key down " increase " " " " "
cursor key left " decrease " " " 20 "
cursor key right " increase " " " " "
a key will allow you to choose another address first a box comes down in zone 10, pressing the cursor down key skips the boxes or press the RETURN key to finish.
The address can be entered as decimal number from 0 to 65535 or hex 0000 to FFFF provided the # is placed before the number.
The Ram page number runs from 0 to 31, 32 is taken as being the Rom.
The printer can be set on by pressing cursor key left or off by pressing cursor right.
r this will toggle address displacement by increasing the address by 16384.

Pressing any key except h a r & cursor keys will bring you back to monitor mode.

DISSASEMBLER MODE

The top left hand corner shows the Ram page and addresses like memory mode(see command m

d enters super fast dissasemble mode in zone 7

cursor key up moves dissasemble back one opcodes intelligently, its a little bit slow and the opcode will not always be what it should , however
cursor key left moves dissasemble back 9 opcodes intelligently, it takes a while but mostly gives the correct opcodes.
cursor key down moves dissasemble forward one opcode.
cursor key right moves dissasemble forward 9 opcodes.
. will smoothly scroll dissasemble forward.
F3 function key F3 same a full stop above but scrolling is slower.
h toggles decimal and hex display, and vise versa
r toggle Ram address displacement, by increasing the address by 16384 bytes.
d toggles Dissasemble mode,bytes & Ascii listings
a enters address mode, similar to command m
p poke mode,the address at the 3rd line, is taken as the poke start address.You can enter decimal or hex (if # is placed before hex number), each number must be seperated by a comma, numbers in the range of 0 to 255 are treated as single bytes 256 to 65535 are treated as double bytes.
Strings can be poked if they are enclosed in quotes (").For example 20,#56,16384,"sam",#FFFF
s search mode, enter the bytes like the poke mode above. After a few secounds,once the search has scanned 256/512K of memory, it will display the searched text, pressing n will scan for more of the same text, if any left.
RETURN gets you back to monitor mode.

TESTING MACHINE CODE

F0 This will execute the one opcode at PC Address

F1 This will skip " " " " " "

F2 This will execute all opcodes one after the other from the start of the PC Address.
Two opcodes are executed per secound.
Press RETURN to stop machine code running.

F3 This will execute all opcodes at a rate of 120 opcodes per secound, however the registers, dissasemble etc are not displayed, but the main opened screen used by basic is displayed useful to examine programs that print to the screen.
Press RETURN to stop machine code running.

The function keys F0, F1, F2 will update the screen on how each opcode effects the registers, memory, flags etc If a set of opcodes should try and print to the screen using the Rom routines, then instead of printing over the monitor screen, it will print on an invisible screen you can view this screen by typing the command scr, to see what text has been printed to that screen.

The function key F3 will display the invisible screen at all times, however the registers and flags etc are not displayed, but are updated, so when you wish to stop the machine code program you want to test, and go back to monitor mode you can see what has happened to the registers etc.

During the execution of opcodes CPDR CPIR INDR INIR LDDR LDIR OTDR OTIR you may find it takes a long time to finish the opcode, upto 5 secounds if you are moving a lot of bytes in memory.

The following opcodes are not executed, but are skipped
DI EI HALT IM0 IM1 IM2 LD I,A LD R,A.

Any Rom addresses that are CALLed, JUMPed to etc are executed at the Sams normal speed, no tests are made in the Rom, for RSTs, JP 0, over popping of the stack etc. When the Rom routine you CALLed etc finishes SC_MONITOR takes control again.
All CALLs and JUMPs to 259 in the Rom will execute normally with the DW address following the opcode.

Some RST opcodes are trapped, and will force you to go back to monitor mode.
RST 0 will be trapped if Port 250 holds Ram page 0,this will prevent the Coupe resetting.
RST 8 all DB bytes after the RST 8 will not execute and you will return to monitor mode.
RST 40 is also trapped.

If when running a test program, the message :-
BREAK STOP Press RETURN to Continue appears then a Breakpoint has forced the program to stop running as set up by command brk, sreg, spreg or by RST trapping.

If the message :- POP STACK OVERFLOW Press RETURN to Continue appears then no more values may be taken off the stack, it means you have POPed more values off the stack than you have put on it.

EXAMPLE TEST PROGRAM

The program below will print the message "Sam Coupe" to the top left hand corner of the screen, using SC_MONITOR we are going to test out this program to see how it works and test for errors.

```
32768 175                 XOR  A         this will clear
32769 205 078 001         CALL #014E     the screen.
32772 062 254             LD   A,254     this will set up
32774 205 018 001         CALL #0112     so text goes to
32777 062 083             LD   A,"S"     the screen.
32779 215                 RST  16        Print character
32780 033 029 128         LD   HL,text   held in A reg.
32783 126          nextch:LD   A,(HL)    this loop prints
32784 167                 AND  A         all text from HL
32785 040 006             JR   Z,reset   until character
32787 229                 PUSH HL        is a byte zero.
32788 215                 RST  16
32789 225                 POP  HL
32790 035                 INC  HL
32791 024 246             JR   nextch
32793 195 000 000  reset:JP    0         this will cause a
32796 201                 RET            crash or will it?
32797 097 109 032         DM   "am "     text for printing
32800 067 111 117         DM   "Cou"
32803 112 101             DM   "pe"
32805 000                 DB   0         text end marker.
```

If you see a black square with a letter R in, this means press the ⏎ETURN key.
First type CLEAR 32767:LOAD "moncode" CODE 32768 ⏎
Then CALL 32768 ⏎ to get into monitor mode.
The code above will run in Ram Page 3 at address 65536
So the first thing we need to do is set up the High memory Port 251 to equal 3, Type 251=3 ⏎
Type d ⏎ this will take you into dissasembler mode.
Press key p this will enable you to poke bytes into memory, to enter the program above into memory type :-
175,205,78,1,62,254,205,18,1,62,83,215,33,29,128 ⏎
You should now see part of the listing above appear.
Press the cursor down key seven times, address 32783 should appear at the top of the dissasemble.
Press key p to go into poke mode for more bytes, type :-
126,167,40,6,229,215,225,35,24,246,195,0,0,201 ⏎
the secound part of the above routine should appear.
Press cursor key right, cursor key down, then
Press key p once more for the last part, type:-
97,109,32,67,111,117,112,101,0 ⏎
The dissasemble may look a bit strange, thats because the pokes you just entered are characters, so press key d three times to see the text, you should now be back in dissasembler mode. Press ⏎

Type pc=32768 ⏎ this sets the start address for the program to test.

type scr ⏎ this will show the invisable screen it should be blank, Press ⏎

Press function key F0 four times, this will run the first four opcodes, you should see the registers change.
Press function key F0 two times to run the opcodes :-
LD A,83 RST 16, this will print the letter S to the invisable screen, type scr ⏎ to view the screen.
Press ⏎ to get you back to monitor mode.

At this stage you should be on address 32780 which will print out a string of text, to show you an example of a Stop Break Register type sa=32 ⏎ you should see the number 32 appear in zone 5.
Press function key F2 to run the progam opcodes slowly and automatically without the need to press any key.
After a short while the message BREAK STOP Press RETURN to Continue. This means that the A register matches with the STOP register A in zone 5. Press ⏎
type scr ⏎ to view the invisable screen, then press ⏎
type csaf ⏎ to cancel the detection if A reg=32.
Press function key F0 twice, so that you are on opcode, PUSH HL, Press function key F0, you should see 32799 on the stack in zone 8, Press the function key F0 twice again,now the 32799 should have been POPed off the stack
type pc=32789 ⏎ so that 32789 POP HL appears.
Press function Key F0, the message POP STACK OVERFLOW Press RETURN to Continue should appear, this came up as there were no more values to be POPed from the stack.
Press Function key F1 to skip the instruction.
type brk0,32783,3 ⏎ this will set up a breakpoint, so if PC should equal 32783 with Port 251 equal to 3 the program will stop running.
Type brk ⏎ to view breakpoints.
Press function key F2 to run the rest of the program.
You should get the message BREAK STOP Press RETURN to Continue, as the address of PC is 32783.
Type brk0 to cancel the Breakpoint address.
Type scr ⏎ to view screen, should see the message Sam
Press ⏎ to get into monitor mode, then press the function key F3 the invisable screen will appear for a tiny fraction and you might just see the message Coupe appear on the screen.
The message BREAK STOP Press RETURN to Continue sould appear when address 32793 JP 0 should be at the top of the dissasemble listing, this error has been trapped if it were allowed to run it would cause the sam to reset.
Press Function key F1 to skip this instruction.
If you type scr ⏎ you should see the rest of the message to the screen "Coupe".
Press ⏎ then press the function key F0 for the last instruction RET, this should display the BREAK STOP error message with RET at the top of the dissasemble listing to show you that the routine has finished.

HINTS AND TIPS

The golden rule on reducing the number of crashes and resets and unexplained problems in large programs, is to write all individual small routines and test them out for ever type of error that could possible happen, including what happens if the stack changed, or if the ram page switched, does it need a large stack, has all the pushes and Pops been balanced, what happens if the interrupt are enabled or disabled, test everything you can.
Then fully document the small routine what registers are needed, what does the routine exactly do, and what registers are corrupted when you return from the routine

I can tell you from personal experience of writing 120K of machine code so far on the Sam, it pays.
Quite often when an error occurs in a large program it can take days to trace down bugs, going through the whole code to find exactly what has gone wrong.

RAM PAGES AND ADDRESSES

256K Sam		512K Sam	
0	16384	16	278528
1	32768	17	294912
2	49152	18	311296
3	65536	19	327680
4	81920	20	344064
5	98304	21	360448
6	114688	22	376832
7	131072	23	393216
8	147456	24	409600
9	163840	25	425984
10	180224	26	442368
11	196608	27	458752
12	212992	28	475136
13	229376	29	491520
14	245760	30	507904
15	262144	31	524288

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.

BINARY System of arithmetical notation using 2 digits - 0 and 1. Upto 8 0' and 1's are used to represent an 8 bit number.

BIT Binary digit, 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.

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

HEX System of numerical notation using 16 digits, the numbers 0-9 and the letters A-F, frequently used in computer programming.

LOW/HIGH A number from 0-255 can be represented in one byte. A number from 256-65535 can be represented in two bytes in the form of :-
byte 1 (the least significant byte or LSB or its Low form)
byte 2 (the most significant byte or MSB or its High form).
A register such as HL has H as the MSB and L as the LSB

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.

OPCODES A machine code instruction built up of upto 4 bytes.

RAM PAGE Block of 16384 bytes. The Z80 processor can address only 65536 bytes at any one time.
Sams 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.

ADVANCED PROGRAMMERS

On the Disc are 2 other files which will be of use to technically Advanced users, who don't use the standard screen as used by Basic for there graphics.

On loading up the file "softwrite", you can alter the screen Ram page the monitor has to use as it's secound screen anywhere in memory.

Also when you come to run the machine code program to test using F3, instead of viewing the standard basic screen, you can define any Ram page screen in any screen mode.

ENHANSED Version 1. 2 Info

New Improvements have been made with Screen printing increased by 11%, bugs ironed out, and a new feature for more comprehensive breakpoints.
The new option on SC_MONITOR is a return to basic before each opcode is executed, so when you use Function keys:-F0,F2,F3 a return is made to basic to run a certain line before each individual opcode is executed giving a more flexible way to test your piece of machine code you would like to test.

For example suppose you have a machine code routine that occupies page 1 in address area 32768-33000, the routine does not make any calls to the rom, nor does it switch pages or make any calls or jumps etc outside 32768-33000 address area. When you call this routine by CALL 32768 it seems to crash, you have checked the code and you cannot find anything wrong with it, this type of crash is usually caused by the routine jumping out of the area 32768-33000.
If SC_MONITOR returns to lets say basic line 10000 you could program the basic to check if PC and the ports don't go outside the address,port area, such as :-

```
10000 IF PEEK (m+7868) <> 1 THEN GOTO 10500
10010 IF PEEK (m+7869) <> 1 THEN GOTO 10500
10020 IF DPEEK (m+7895) <32768 OR DPEEK (m+7895) >33000
      THEN GOTO 10500
10030 RETURN
10500 SCREEN 1: PRINT "ADDRESS/PORT OUT OF RANGE":PAUSE:
      RETURN
```

Note m is the start of SC_MONITOR loading address, now the registers are all stored in an area 7868 bytes from the start address are are constantly updated as each opcode is executed. Line 10000 and 10010 tests if Port 250 or Port 151 holds 1, Line 10020 test PC is in range, Line 10030 Returns you back to the Monitor program, Line 10500 prints the warning message that PC or Ports have jumped out of range, i.e this is where the bug lies in the program.
Here is another example look at the following ocodes:-

```
32768 33  0  64  LD   HL,16384
32771 62  199    LD   A,199
32773 50  3  128 LD   (32771),A
32774 201        RET
```

This is a sample little routine that does not do much if you CALL 32768 it runs and return to basic, but if you try to CALL 32768 again it will crash, quite mysterious, this type of crash is usually caused by a routine corrupting itself. One way to find out which bytes have been overwritten is to verify the routine against your source code, this will tell you which byte has corrupted in the above case its address 32771 which has changed from byte 62 to 199 (the opcodes RST 0).
So line 10000 could be programmed as :-

```
10000 IF PEEK 32771 <> 62 THEN SCREEN 1: PRINT "BYTE COR
      RUPT": PAUSE: RETURN
10010 RETURN
```

Another example of using the flexible return to basic is to have a trace function whereby in some part of memory you store the Port 250,Port 251 and PC Address for each opcode that is exceuted. So when the routine you wish to test locks up, just press the NMI button to get out of the lock up, as you have the adresses stored in memory you can dissenble the last few executed addresses to find out why sam locked up. If however instead of a lockout your coupe reseted, the trace is no good as memory is cleared from memory, unless you have the SC_AUTOBOOT chip this does not clear memory, so you can find out where your routine caused a reset.
On the Disc you have is a basic program called "trace" which will provide a trace function provided that a return to basic is made to line 10000.
Also on disc is a stand alone dissasembler which can be loaded into the start of a Ram page address such as 32768,65536 etc It is 10700 bytes long, lines 10 onwards in the trace basic will dissasemble the data held in the trace table.

The Registers held in SC_MONITOR are at the start address + 7868 bytes after:-

```
If using moncode  LET r=m+7868   (note m is the start
If using moncode+ LET r=m+768+21  address of Monitor)
Port 250 r     (So PEEK r will give you the Port 250)
Port 251 r+1       PEEK (r+1) will give you Port 251)
Port 251 r+2
HL'      r+6
DE'      r+8
BC'      r+10
AF'      r+12
IX       r+14
IY       r+16
HL       r+18
DE       r+20
BC       r+22
AF       r+24
PC       r+27
SP       r+33  (current SP address)
SP main  r+36  (as set by SP command)
```

When using SC_MONITOR zone 3 the AF register shows A as the low byte and F as the high byte, if you would like it the other way round POKE Monitor start address+8212, 107,90,85 for moncode+ start address+8233,107,90,85.

ASSEMBLER TIPS

If you do not like the character set on SC_MONITOR and you would like to change it and you have a copy of SC_ASSEMBLER 256K 1.2 or 512K 1.0 then do the following-

Set up the configured Assembler for the character set you would like to use, follow on until you get to the part of saving the configured Assembler to disc, Press ESC. Type NEW plus RETURN and then Type:-
CLEAR 29999: POKE 30000,mem$(457144 TO 457144+1535):LOAD "moncode" CODE m: POKE m+14592,mem$(30000 TO 30000+1535)
Then just SAVE "moncode" CODE m,32768.
If you are using the 256K Assembler change 457144 to 96696, the variable m is where the monitor code starts.

When using SC_ASSEMBLER and you wish to save to a ram disc instead of just Drive 1 or 2 then on the 512K version POKE 462996,56,228,254,56,48,224.

MERGING MONITOR & ASSEMBLER

If you have SC_ASSEMBLER 512K version and you would like to have SC_MONITOR in the same memory so you can switch between the two using the 512K version only but don't mind that source Banks 7,8,9 cannot be used do the following :-

Load in SC_ASSEMBLER 512K,Type CLEAR 29499 to allow more space to Replace the Basic lines to :-

```
16 MODE 3: POKE SVAR 50,1: CLEAR 29499
17 DPOKE 294912,65535: DPOKE 327680,65535: DPOKE 360448,
   65535: DPOKE 16426,16384: CALL 16384
20 ON ERROR STOP: LET n=PEEK 23701: RESTORE 25: FOR a=0
   TO n: READ d: LET k=PEEK d: NEXT a
21 IF k=81 or k=113 THEN STOP: REM key q
22 IF k=66 or k=98  THEN STOP: REM key b
23 IF k=88 or k=120 THEN GOTO 30: REM key x monitor
24 STOP
25 DATA 23699,23700,23693,23694,23695,23696,23697,23698
30 ON ERROR GOTO 1
31 DPOKE 294912,60915: CALL 294912
1001 LOAD "moncode" CODE 294912
```

When you use the Assembler pressing q or b then RETURN will drop you to basic and STOP, x and RETURN will enter the Monitor pressing q and RETURN will go back to the Assembler.

NOTES

NOTES