Editor: G.A.BOBKER of ZX-GUARANTEED

At the time of printing MDX21 the 128K SPECTRUM 2 by the Amstrad people, has still not appeared. It is extremely unlikely that it will allow an Interface 1 to be housed under it.It is claimed it will simply be a Spectrum 128 in an Amstrad keyboard with a tape recorder built-in. Bearing in mind that companies such as Ocean still continue to churn out difficult to load tapes, therefore it is necessary to have several tape machines to hand. The "old" Spectrum 128K by Sinclair would appear to be a better buy AND it will be much cheaper...obviously this is assuming that Amstrad's version will be fully identical to the Sinclair version. This is to be followed in the near future with a method of connecting it to an Amstrad Disc system.It is interesting to note that an OPUS Disc drive is fully external to a Spectrum, yet claims to use no Bytes (altho' it appears to use a temporary Buffer of 512 Bytes) An Amstrad Disc built into an Amstrad machine by their engineers who "should" know their system, uses approx 1200 Bytes plus a tempory Buffer of a massive 4K. This doesn't say much for the expertise of the programmers. (In reply to a letter I sent them about this and the atrocious ROM layout, they said "our machines show how the "professionals" write their programs"...NO COMMENT)

```
*****************************************************************
* This issue of Microdrive-Exchange is devoted entirely to how *
* to move ROM Load/Save routines higher in memory & modify to  *
* obtain some of the effects the "professionals" use. All the  *
* routines include Demo programs for non-machine code members. *
*****************************************************************
```

JUMPING TO MACHINE-CODE

If you intend going into a machine-code routine & staying there until the Spectrum is powered off (as is the case with games), then any of the methods could be used. If program has to Return to Basic, then study the Return actions to determine which is the most suitable to your requirements.

1. RANDOMIZE USR 30000 This will jump to address 30000 & then starts obeying whatever is in 30000 onwards as machine-code. A return to Basic is done without problem by including a RET instruction at a suitable point in the Code.
2. RUN USR 30000 This is the same as the above except that most programmers say you should ONLY use it if you DON'T intend to to return to Basic (& don't say what would happen if you do)? I have used RUN USR xxxxx without problems, but suggest that it may be more prudent to only use it if staying in Code.

```
3. PRINT USR 30000        These four command are different in that
4. LET X=USR 30000        if you make the machine-code return back
5. GOTO USR 30000         to Basic, the last value present in the
6. GOSUB USR 30000        BC register pair can be used.
```

PRINT USR would print the value. LET X=USR will put the value in X. This can be useful in games part in Basic with machine-code being used for scroll, etc. Value in X being used to check if a "collision" or whatever, or simply to pass info from machine code back into the Basic. The GOTO or GOSUB is rarely used but means that on return you can go to a line number or sub-routine determined by value the machine-code puts into the BC register pair. In reality these should be used with care as checks I did tended to give problems with variables.

A test program to check out the actions is shown on next page.

```
 10 PRINT AT 0,0;
 20 DATA 1,50,0,201    <---This Loads BC with 50 then Returns
 30 FOR J=35000 TO 35003: READ A: POKE J,A: NEXT J: STOP
 50 PRINT "IT WENT TO LINE 50": STOP  <---This MUST be Line 50
100 PRINT USR 35000: STOP
200 LET X=USR 35000: PRINT "X=";X: STOP
300 GOTO USR 35000
400 GOSUB USR 35000
```

RUN program then enter in turn, GOTO 100, GOTO 200, GOTO 300 or GOTO 400 to try out the actions.

Most of the earlier games tended to go into machine code by using the PRINT USR. Main reason was simply that it is the only one quoted in the Sinclair Manual.

ALTERING THE SAVE/LOAD ROUTINES

The methods explained are long to write, but Demo programs have been included to ensure that even without machine-code knowledge you can try out all the actions easily.

The first requirement is to move the relevant part of the ROM to higher in memory. By altering the SAVE/LOAD action in this HIGH-ROM it can be made to do weird things. In this issue we will see how to alter Loading Border and/or alter Speeds. For convenience we will move the part of ROM we require to be 50000 Bytes Higher in memory. This means that instead of using a LOAD "" which causes a Jump to 1366 in ROM, we use instead RANDOMIZE USR 51366 It is NOT that easy as several machine-code addresses have to be changed to allow for the higher ROM addresses. A Dissassembler is an obvious asset as then you'd be able to see addresses which need to be changed. Method explained here is for Headerless programs ONLY and Returns to Basic each time. If you want to do more than this, the book SPECTRUM ROM DISSASSEMBLED by Dr Ian Logan is a must. (Difficult to buy now, but try W.H.SMITH'S or BOOT'S). The total ROM part we require for the SAVE/LOAD is address 1218 to 1540 inclusive. Program to move this to high memory & modify the addresses is as follows;

```
 10 LET A=51218
 20 FOR J=1218 TO 1540       The first 13 POKEs are intentionally
 30 POKE A, PEEK J           put BEFORE this relocated SAVE/LOAD.
 40 LET A=A+1: NEXT J
 50 POKE 51205,221: POKE 51206,33: POKE 51207,0:  POKE 51208,64
 60 POKE 51209,17:  POKE 51210,0:  POKE 51211,11: POKE 51212,62
 70 POKE 51213,255: POKE 51214,55: POKE 51215,195:POKE 51216,166
 80 POKE 51217,200: POKE 51254,40: POKE 51255,200:POKE 51276,87
 90 POKE 51277,200: POKE 51292,117:POKE 51293,200:POKE 51320,100
100 POKE 51321,200: POKE 51336,78: POKE 51337,200:POKE 51389,55
110 POKE 51390,201: POKE 51404,51: POKE 51405,201:POKE 51411,51
120 POKE 51412,201: POKE 51425,55: POKE 51426,201:POKE 51436,55
130 POKE 51437,201: POKE 51483,51: POKE 51484,201:POKE 51493,26
140 POKE 51508,55:  POKE 51509,201
```

(The 201's are part of addresses & are NOT Return instructions).

Save to microdrive by SAVE*"m";1;"HIGHROM" CODE 51205,336

Machine-Code users please note that the first 13 Bytes are as shown on the right. Note that a POKE 51216,166 gives JP as shown, and POKE 51216,18 makes it JP 51218

```
LD IX,16384
LD DE,6912
LD A,255
SCF
JP 51366
```

In order to use the SAVE or LOAD routines it has to be told the Length & Start location (and Code type) for the program that we intend to SAVE/LOAD. This means make DE=Length, IX=Start loc' and set A register=Type. The extra 13 Data Bytes POKEd in will initially set IX=16384, DE=6912, A=255 so we can try routine on a screen$ picture.Note that in all checks we will Load a screen$ AFTER its Header.

The routine is only a copy of the Save/Load and should operate as normal AND is initially set to act as a Load routine. Check this as follows; (and DON'T use CLS after picture appears)

1. Take an old game with a normal Screen$ and set the tape to be just AFTER the Header of the Screen$.
2. Enter RANDOMIZE USR 51205 then play in the rest of Screen$. This should Load and picture will appear on the screen.
3. To check the Save action the HIGHROM has to be switched to Save by entering POKE 51216,18.
4. Put a Blank tape in recorder and set to record. Enter; RANDOMIZE USR 51205 and Headerless Screen copy will Save out.
5. This tape can be checked by loading in the original Header of the Screen$ from original tape, then play in tape just made.

FANCY COLOURED BORDER (or NO Border)

(Whilst it is possible to change the Border Loading lines to be any pair of colours it is rather long to explain and present. If any members do experiments on this and come up with short action which can be tagged onto programs given in this issue, then they will be printed in a later issue).

LOAD Border colours can be changed by entering; POKE 51536,x with x being a value of 0 to 7 (7 gives NO Border). Set HIGHROM to Load by entering POKE 51216,166. Enter RANDOMIZE USR 51205 and again play in a Headerless Screen$ to see effect.

The program below will Load in a screen$ picture from a normal speed program (Note: As stated, this will ONLY Load in Headerless, therefore, play in the Screen$ AFTER its Header).

```
10 LOAD*"m";1;"HIGHROM" CODE 51205
20 POKE 51536,7      <---------Change this to get Border effects.
30 PRINT "Play in a Screen$ AFTER its Header"
40 RANDOMIZE USR 51205
```

After RUNning this and playing in a screen$ (WITHOUT the Header) try modifying by changing the Poked value in line 20 to be a value between 0 to 6.

Type in the program below;

CHECKER PROGRAM

```
 10 LOAD*"m";1;"HIGHROM" CODE 51205
 20 INPUT "Border Number (7=none) ";B
 30 POKE 51536,B
 40 INPUT "1=LOAD  2=SAVE ";X
 50 POKE 51216,166: IF X=2 THEN POKE 51216,18
 60 INPUT "Memory Address ";A
 70 POKE 51208, INT (A/256): POKE 51207, A-PEEK (51208)*256
 80 INPUT "Length ";L
 90 POKE 51211, INT (L/256): POKE 51210, L-PEEK (51211)*256
240 RANDOMIZE USR 51205  <-----This is line 240 for later use.
```

CHECKER program will simplify the using of HIGHROM (and will be modified later for Speedy SAVE and LOAD). As a check, type it in (then Save it to microdrive), RUN and enter 1 to say Load. Enter 7 for Border Number. Enter Memory Address as 16384 and Length as 6912. Now play in a Screen$ WITHOUT Header.

SPEED SAVE/LOAD

The ROM Save/Load routine includes various delays and checks to ensure these are at the designated speed of 1500 Bauds. The very maximum the Spectrum can handle is approx just over TWICE normal speed. The highest speed ever used in a game was with Kokotoni-Wilf which was at twice normal.Usually the fast games are at 1.5 times normal in order to ensure they will Load O.K. In practise, to adjust the speed, we alter 4 timing Bytes in the SAVE & 5 in the LOAD routine. Once again this is only for using with Headerless programs, but could be developed further for normal program types (by you).

Four speeds can be used by using the Speed Table, or by adding these extra lines to the CHECKER program. Speed 0 is the normal Speed of 1500 Bauds. Speed 4 is just over twice normal.

```
100 INPUT "Speed Number ";S
110 IF S=0 THEN GOTO 220
120 LET S=1500/(S*750+1500)
130 POKE 51272, INT (59*S)
140 POKE 51305, INT (66*S)
150 POKE 51311, INT (62*S)
160 POKE 51326, INT (49*S)
170 POKE 51446, INT (176*S)
180 POKE 51479, INT (178*S)
190 POKE 51487, INT (203*S)
200 POKE 51512, INT (22*S)
210 PAUSE 0
220 IF INKEY$="c" THEN CLS
230 GOTO 20
```

SPEED TABLE

	ADDRESS	0	1	2	3	4
S	51272	59	39	29	23	19
A	51305	66	44	33	26	22
V	51311	62	41	31	24	20
E	51326	49	32	24	19	16
L	51446	176	117	88	70	58
O	51479	178	118	89	71	59
A	51487	203	135	101	81	67
D	51492	176	117	88	70	58
	51512	22	14	11	8	7

RUN program and Load in a Screen$ but tell it to Load into 30000 (as we need screen area). When Loaded, press a key and program will re-run. Tell it to Save out Address 30000, Length 6912 at Speed 3, and Save to a tape. When Save out complete, press enter key and this time tell it to reload your copy tape into Address 16384 at Speed 3. Picture will then Load & appear at high speed. Note that for practical use your program would have to Load in the high speed Loader (the LOADer part is in address 51366-51540 inclusive) and needs to be arranged to set the IX, DE, etc. This would then be made to Load in the Fast part(s). To be devious to make copying of your programs awkward, make the speed only just slightly Faster and with a weird coloured, or no Border.

If you have a Fast Headerless program you want to convert down to normal speed, HIGHROM could be used by loading in at CORRECT speed (if you don't know speed, try different speed numbers till it Loads in),then Save it at normal speed. BUT, it ONLY works if Code type is 255. Code type can be altered by POKEing 51213 to the code of the Headerless part you want to Load in.

HIGHROM will be used again in a future issue.
