MICRODRIVE EXCHANGE

Editor: G.A.BOBKER of ZX-GUARANTEED

When the Microdrive was first launched, it was hailed as the cheapest method of mass storage which offered a fairly fast access time. For £100 it WAS good value for money. The OPUS Disc system has now dropped in price, and is available in BOOT'S for £65. The Microdrive system IS very reliable, even allowing for the occasional "loss of memory". Given the choice, the first time buyer would be a fool not to get the OPUS. Major failing of the Microdrive system was the initial high cost of the cartridges, altho' they dropped eventually to £2. OPUS 3 1/2 " Disc are £6 from BOOT'S but will be about £2 each in the future as these Discs are the new business standard. One problem still remaining is the difficulty of transferring programs to 'drives. The OPUS no doubt will see a repeat of the Microdrive spate of tape transferrers. More about this in the next issue.

<u>CORRECTIONS TO MDX21</u>. Corrections to HIGHROM routines are;
Bottom of page 1;
Program line 60. The POKE 51211,11 should be POKE 51211,27
Program line 120. Two addresses are wrong. Change as follows;
POKE 51425,55:POKE 51426,201 to be POKE 51426,55:POKE 51427,201
Program line 130. Change address at end of line & add in a POKE;
Was POKE 51493,26 change to be POKE 51494,26:POKE 51495,201

On page 4, add in a line 195 thus; 195 POKE 51492, INT (176*S)

PAPERBOY TO MICRODRIVE

This is actually Loaded at normal speed but with a fancy Border. The built-in Loader has sufficient room to enable it to be modified so that we can Load game section and then Save it out to a tape. Main Block of Code actually starts at 16384 and is 48616 Bytes long. Method used is to modify the original Built-in Loader (which goes into 65000), so that the Code Block Loads into 9472 which means the screen section will be chopped off and gives us room to manipulate the remainder.

1. Type in this program;

```
10 CLEAR 64999: LOAD "" CODE 65000
20 POKE 65012,0: POKE 65013,0: POKE 65014,243: POKE 65018,37
30 DATA 243,62,191,219,254,31,56,249,221,33,0,72,17,212,154,
   62,255,205,194,4,243,62,191,219,254,31,56,249,221,33,0,64,
   17,0,8,62,255,205,194,4,195,0,0
40 FOR J=65025 TO 65067: READ A: POKE J,A: NEXT J
50 RANDOMIZE USR 65014
```

2. RUN the above, then play in your PAPER tape from the start. (It will ignore the Basic on it). The main Block of Code will Load in and the screen will eventually show coloured squares. STOP the tape when the Code block has Loaded as there is an extra part at normal speed at the end.
3. Place a Blank tape in recorder and set to record. Press the Enter Key and a 39636 Byte Headerless Block will be Save out. When Saved, allow a few seconds gap, then press Enter Key again and a second Block of 2048 Bytes will Save out. The Spectrum will NEW after second Block Saved out.
4. Type in this program which will transfer last part of the PAPERBOY tape to Microdrive;

```
10 DATA 221,33,0,99,17,0,4,55,62,255,205,86,5,201
20 FOR J=23300 TO 23313: READ A: POKE J,A: NEXT J
30 RANDOMIZE USR 23300
40 SAVE*"m";1;"PB1" CODE 25344,536
```

5. RUN the program just typed in, then play in last part of the PAPER tape and it will Transfer it to Microdrive.
6. The two Headerless Blocks on the tape saved previously, are now transferred to Microdrive using program below;

```
 10 DATA 221,33,0,99,17,212,154,55,62,255,205,86,5,201
 20 FOR J=23300 TO 23313: READ A: POKE J,A: NEXT J
 30 RANDOMIZE USR 23300
 40 SAVE*"m";1;"PB2" CODE 25344,39636
 50 RANDOMIZE USR 23300
 60 SAVE*"m";1;"PB3" CODE 25344,2048
```

7. RUN the above and then play in the Headerless-Files you Saved out before. Stop the tape when Microdrive starts running.When it has transferred, play in next block.
   Note that the program IS correct and the same DATA block is used to Load in each of the Headerless-Files. (It doesn't matter that second Block is only 2K long as the "Loader" will simply return back to Basic when the Block has loaded in.
8. Type in this Basic Loader for the game;

```
 10 CLEAR 25343
 20 LOAD*"m";1;"PB1" CODE 65000
 30 LOAD*"m";1;"PB2" CODE 25344
 40 LOAD*"m";1;"PB3" CODE 16384
 50 DATA 243,49,0,88,33,0,64,17,0,91,1,0,8,237,176,195,227,186
 60 FOR J=18432 TO 18449: READ A: POKE J,A: NEXT J
 70 RANDOMIZE USR 18432
```

Save this by; SAVE*"m";1;"PB" LINE 10

ROUTINES USING HIGHROM

It was hoped to include a routine to put a counter or action on the screen whilst a program was Loading. Unfortunately the use of printing to screen routines when loading seriously affects the timing ang would necessitate radical changes to the HIGHROM.

- - - - - - - - - - - - - - -

USING HIGHROM TO LOAD BACKWARDS

Loading in programs such that the screen is drawn from bottom to top is easy now that we have the HIGHROM routine. Normally the ROM when Loading/Saving, keeps Incrementing IX (means add 1) to get the next location to put the Byte into, & Decrements the DE register (means subtract 1), until DE, which starts by holding the length of the program, becomes 0. In the HIGHROM these are the INC IX and DEC DE a Dissassembler would reveal. By changing these to be DEC IX and INC DE, and presetting them initially, we can easily Load in programs backward. Load in the CHECKER prog from MDX21, and add the following lines;

```
85 INPUT "1=NORMAL  2=BACKWARDS ";Z
86 POKE 51324,35: IF Z=2 THEN POKE 51324,43
87 POKE 51475,35: IF Z=2 THEN POKE 51475,43
88 INPUT "Set tape, then press any key ";A$: PAUSE 0
```

RUN modified CHECKER and answer the questions it prints so as to
1. Load into address 16384 with length 6912, and NORMAl.
2. Play in a screen$ without its Header (Do NOT use CLS).
3. Enter GOTO 20 to restart without losing the picture.
4. This time answer questions so that it will Save from address 23295 with the length 6912 and BACKWARDS.
5. When Save complete, enter CLS.
6. Enter GOTO 20 and this time Load using address 23293 (yes it is correct address), length 6912 and BACKWARDS.
7. Play in the Backward screen$ file you just saved and it will load in from bottom of the screen.

- - - - - - - - - - - - - - -

HIGHROM AS A BYTE COUNTER FOR FILES, ETC

By making the INC IX inoperative, by making it always zero, the routine can be made to effectively NOT load in a program, and by setting DE to a high value at the start, them checking it at the finish, the HIGHROM can be made to act as a Byte Counter. Can be very useful for Headerless-Files to find their length. Modified program is then Saved as "COUNTERc" (after beginning part has been moved to be just before the part we will be using).

```
 10 LOAD*"m";1;"HIGHROM" CODE 51205
 20 POKE 51207,0: POKE 51208,0        <---Sets IX=0
 30 POKE 51210,1: POKE 51211,255      <---Sets DE=65281
 40 POKE 51474,0: POKE 51475,0        <---Disables the INC IX
 50 POKE 51215,8: POKE 51216,205:     <---Changes JP 51366 to CALL
  POKE 51217,169: POKE 51218,200         51366 & inserts EX AF,AF'
 60 POKE 51219,213                    <--PUSH DE
 70 POKE 51220,193                    <--POP BC
 80 POKE 51221,201                    <--RET
 90 LET A=51349: FOR J=51205 TO 51221
100 POKE A,PEEK J: LET A=A+1: NEXT J
110 SAVE*"m";1;"COUNTERc" CODE 51349,244
```

Since only the Loader part of the routine would be used, only the Loader part is Saved by line 110.

Suitable Basic for the Counter is now;

```
 10 LOAD*"m";1;"COUNTERc" CODE 51349
 20 PAPER 6: CLS: PRINT TAB 10; "BYTE COUNTER"  <---Put Title in
 30 LET X=USR 51349: PAPER 7                        INVERSE.
 40 PRINT AT 10,14; "     "
 50 PRINT AT 10,14; 65279-X
 60 GOTO 30
```

RUN program, then play in a screen$ WITH its Header. The Header of a program is ALWAYS 17 Bytes and 17 will be print on screen. Let tape continue to load, and after the screen part Loaded in, the value 6912 will appear. If when you use this program with a Screen$, the answer is greater than 6912, it will mean that the game/program actually loads some Bytes into the Printer Buffer & is most likely a "hidden" routine to load in another Headerless-File from off a tape.

A study with a Dissassembler will reveal that Bytes 51366-51368 aren't used at all, but have been left in so you can see the subtle difference necessary to enable the Loader to accept ANY type of File.

"RECLAIM" ROUTINE

The Reclaim routine can be used to delete several lines of Basic from a program. To do this we have to set the DE reg to point to the First Byte to be "reclaimed" and set HL to the first byte to be left alone, then JUMP to Reclaim routine in the ROM. (Note that you should use a JUMP to it and NOT by the usual CALL; altho' doing a CALL would often work). As an example, the following program will "reclaim" lines 20-50.

```
 10 CLS #
 20 PRINT
 30 PRINT
 40 PRINT
 50 PRINT
 60 PRINT
 70 DATA 17,12,93               <---Set DE=First to be reclaimed
```

The CLS # in line 10 has been included to ensure that the MD map is switched in otherwise all the addresses would 58 Bytes lower....very important here as the HL & DE MUST know addresses.

```
 80 DATA 33,36,93               <---Set HL=First to leave alone.
 90 DATA 195,229,25             <---JP to reclaim routine.
100 FOR J=23300 TO 23308        Could be altered to make it also
110 READ A: POKE J,A: NEXT J    Delete itself after use.
120 RANDOMIZE USR 23300
```

If, before running the above, you enter CLS ## then enter as a direct command; FOR J=23813 TO 66666:PRINT J;"=";PEEK J:NEXT J you can then see how the addresses for HL & DE are derived. The values printed on screen would be;

```
23813=0                  Remember that ALL line numbers use 2 Bytes
23814=10                 and are followed by 2 Bytes holding length
23815=3                  of the line. Clearly in this example, the
23816=0                  First Byte to be reclaimed is at address
23817=251                23820
23818=35
23819=13                 You should also find that First Byte to be
23820=0                  left alone is address 23844
23821=20
 ....
 ....
23843=13                 23843 is the End-Of-Line 50. The 23844 is
23844=0                  the start of Line 60 which is therefore the
23845=60                 the First Byte to be left alone.
```

The above method of "looking" at program locations has been printed before but this, and method of splitting addresses up is very important and does tend to give problems.
The address 23820 has to be split into the two Bytes for a DATA statement as follows;
High Byte=INT (23820/256): Low Byte=23820-High Byte * 256
Work these out on a calculator (I.E. get the Spectrum to do it), then check your answer agrees with my figures in the DATA statements. Note also that in the DATA statement they are put in the order Low Byte then the High Byte.

SOFTROM REVIEW

This is the most powerful add-on available. Basically it is 16K programmable memory which can be switched over the ROM area, either to replace the ROM or supplement it. SOFTROM is programmable, which means that it cannot go out of date. Could transfer Sinclair ROM into it and put a copier in the 1170 free Bytes of the ROM. Load a game as normal, press a switch and a Screen shot is saved out just like the computer mags do. Could be used to turn a 128K Spectrum into a GENUINE 48K so that the older games will Run. (The SOFTROM was used in the development of the Interface 007). ZX-GUARANTEED intends to bring out a large range of software for it at realistic prices and will include a Screen Saver, a Copyall Kopykat, a program to turn the 128K into a true GENUINE 48K, Tape-To-Microdrive copier, Dissassembler, etc, etc. SOFTROM is normally £44. ZX-GUARANTEED by special arrangement with EXPANDOR SYSTEMS can supply SOFTROM for just £39.95 complete with all necessary instructions, plus several programs to start you Off, including a Copyall copier.

Postage/Packing in U.K. +£1. Overseas Europe +£2.50. Others +£6 (All cheques/P.O.s should be made payable to G.A.Bobker).
