ENTERPRISE
COMPUTERS LTD

31-37 Hoxton Street
London N1 6NJ
Telephone 01-739 4282
Telex 22717 ENTER G

APPLICATION NOTE NO. 14

SAVING STRING & NUMERIC ARRAYS

To save numeric arrays:

1. First, create your array e.g. 100 dim x (10).

2. Put what you wish to be committed to tape into the array e.g.:

```
100 for x=1 to 10
110 let t(x)=x*x
120 next x
```

3. Open a channel between 1 and 99 for output e.g.:

   open £20:"tape:filename" access output

   The channel number can be in the range of 1 to 254, but some of the channels in this range are used for other purposes such as the graphics screen, etc..

   As well as this, only channels between 1 and 99 are closed automatically after use. The file name can be up to 31 characters long (not spaces) or non-existant. If the file name is left out, it is still important to put the colon after the word 'TAPE:'.

4. Straight after the 'OPEN' line you should have the part of the program that writes to the tape. The writing to the tape is achieved with 'PRINT' statements to the relevant channel. So if your 'OPEN' statement looks like this:

   OPEN £25:"TAPE:FILENAME" ACCESS OUTPUT
   then you should have a print statement like this:
   PRINT £25:VARIABLE(K)
   Where variable (K) is the array you wish to save and K is an element of that variable.

5. Open a channel for input. This need not be the same channel that was used for saving the array. Open the channel like this:

   OPEN £50:"TAPE:FILENAME" ACCESS INPUT

6. Immediately after this line should follow the part of the program that reads the data in:

   INPUT £50:VARIABLE(K)

   This is similar to PRINT £50 except that it accepts data from tape instead of pushing it out to tape.

7. Dealing with strings and string arrays.
A dollar sign should be placed after the variable or array name to be saved:

```
PRINT £50:VARIABLE$(K)
```

This is the same for all string operations and is standard BASIC. You should be careful when saving data intending to be shared between more than one array. Unless you have a specific reason to do so do not put more than one print statement on a line if you are printing to tape e.g.:

```
100 PRINT £50:G$(K),P$(K)
```

This will cause out of data errors when you try to read the data back because G$(K) AND P$(K) will be concatenated.

Therefore it is advisable to use more than one line (two in this example) e.g.:

```
100 PRINT £50:G$(K)
110 PRINT £50:P$(K)
```

When loading in the data file it IS acceptable to have the variables for input joined on one line e.g.:

```
INPUT £50:G$(K),P$(K)
```

This is equivalent to:

```
INPUT £50:G$(K)
INPUT £50:P$(K)
```

```
100   PROGRAM "FILE-CREATION"
110   ! AUTHOR G. MORGAN.
120   ! DATE  27/02/85.
130   TEXT
140   STRING PLACE$(9)*20,CAPITAL$(9)*20
150   PRINT AT 5,6:"PRESS RECORD + PLAY ON CASSETTE RECORDER
      THEN HIT THE SPACE BAR"
160   IF JOY(O)=O THEN 160
170   OPEN £99:"TAPE GEO-FILE" ACCESS OUTPUT
180   FOR X=1 TO 8
190   READ PLACE$(X),CAPITAL$(X)
200   NEXT X
210   FOR X=1 TO 8
220   PRINT £99:PLACE$(X)
230   PRINT £99:CAPITAL$(X)
240   NEXT X
250   CLOSE £99
1000  DATA ENGLAND, LONDON, IRELAND, DUBLIN, USA, WASHINGTON,
      SCOTLAND, EDINBURGH, FRANCE, PARIS, ITALY, ROME, HOLLAND
      THE HAGUE, WALES,CARDIFF
```

N.B. TO MAKE THIS DISPLAY NEATER, TRY CHANGING LINE 130 TO:
TEXT 80

```
100   PROGRAM "FILE-READING"
110   ! AUTHOR G. MORGAN.
120   ! DATE  27/02/85.
130   TEXT
140   STRING PLACE$(9)*20,CAPITAL$(9)20
150   OPEN £10:"TAPE:GEO-FILE" ACCESS OUTPUT
160   FOR X=1 TO 8
170   INPUT £10:PLACE$(X)
180   INPUT £10:CAPITAL$(X)
190   NEXT X
200   PRINT:PRINT
210   PRINT TAB(10);"......COUNTRY......";TAB(40);"......
      CAPITAL......"
220   PRINT TAB(10);"-------------------";TAB(40);"------
      -------------"
230   PRINT
240   FOR X=1 TO 8
250   PRINT TAB(10);PLACE$(X);TAB(40);CAPITAL$(X)
260   NEXT X
270   PRINT TAB(10);"THAT IS THE END OF THIS DATA FILE!"
280   END
```

N.B. TO MAKE THE DISPLAY NEATER, TRY CHANGING LINE 130 TO:
TEXT 80