Personal Computer World

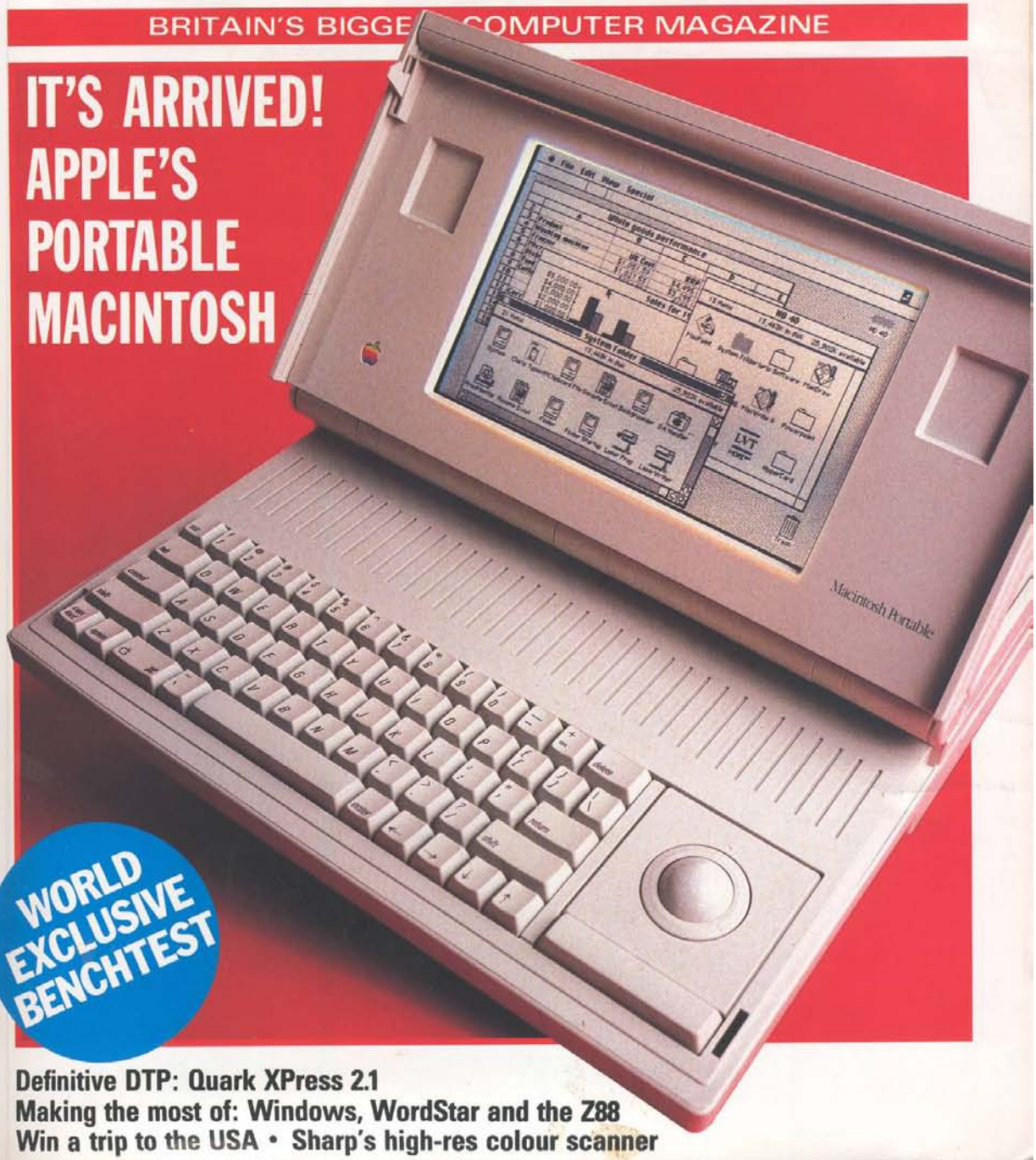
PCs in Russia: Exclusive inside story

US$ 4.50/BFr 13.8/Lire 6400/DKr 38.5/DM 10

October 1989 £1.50

BRITAIN'S BIGGE[illegible] COMPUTER MAGAZINE

IT'S ARRIVED! APPLE'S PORTABLE MACINTOSH



WORLD EXCLUSIVE BENCHTEST

Definitive DTP: Quark XPress 2.1

Making the most of: Windows, WordStar and the Z88

Win a trip to the USA • Sharp's high-res colour scanner

programmer (by using {$I—} and inspecting IORESULT, for example) will cause the procedure 'AllErrorHandler' to be activated. This displays a short message then restarts the program at the main menu to await further user input.

In practice you would wish to give the user more feedback on the nature of the error: one way of doing this is to have the error handler open and read a text file containing explanations of the error, and display the appropriate error message in English. You might also wish to record such errors in a log file for diagnostic purposes.

The program EXCPDEMO.PAS demonstrates some of the more sophisticated capabilities provided by the Exceptions unit. These include the ability to signal exceptional conditions (not necessarily errors) within a program and have a handler deal with them in a way that bypasses the normal block structuring of Pascal.

EXCPDEMO.PAS has been compiled and tested under Turbo Pascal 4. It should work under Turbo 5 without alteration. The source code contains various statements that are commented out — these can be used to vary the effects of some of the handlers set up by the program.

Getting the point in Basic

by TJ Chappell

Acorn's latest version of BBC Basic has improved many of the features of previous versions and added many new facilities. However, the latest version is still limited by a maximum of 9 decimal places and a maximum exponent of only 38. These limits are normally acceptable for most users but what do you do if you require more accuracy or a wider range of numbers?

Until now the only possibilities were to use a scientific calculator and enter the numbers into PRINT statements, or write a machine code program to use the facilities of the floating point unit. Neither of these options offer a complete solution to the Basic programmer.

The solution offered here removes the need to learn how to program the floating point unit directly or press any buttons on a calculator. The program is presented as a function library which can be installed into the machine in the normal way using the INSTALL, LIBRARY or APPEND commands as required. It also relies upon the floating point emulator being installed as well.

As Basic would round numerical variables to suit its own precision, it is necessary that all extended accuracy numbers are stored in string variables. The numbers are all stored in the following format :

```
+/-m.mmmmmmmmmmmmmmmmmm
  E+/-eeee
```

This leads to numbers with a precision of 18 decimal places and a maximum exponent of 9999 (a somewhat larger range than normal Basic and most scientific calculators).

All variables to be used by the system must observe the above format and may be initialised to a starting value using the FNCONST routine, for example:

```
number$=FNCONST("1.234567")
value$=FNCONST("3.101E69")
old$=FNCONST(STR$(old))
```

The FNCONST routine will accept numbers in any of the above forms. Note that in the last example, old$ will contain a maximum of 10 significant figures since that is the limit imposed by Basic.

Having initialised a variable it is then possible to use any of the commands provided by the library. The commands are summarised on the opposite page and comprise two main forms. The first type require only one string variable to be passed to them, examples include:

```
neg$=FNNEG(pos$):REM neg=-pos
sinangle$=FNSIN(angle$):REM
   sinangle=SIN(angle)
inta$=FNINT(a$):REM inta=INT(a)
```

It is possible to include constants that have not been created using FNCONST as long as the format described above is used. The following statements will achieve exactly the same result:

```
pi$=FNACS(FNNEG(FNCONST("1")))
pi$=FNACS("1")
```

Note from the above example that is is perfectly possible to nest the routines: that is, the result of one routine being passed directly to another. There is no limit imposed upon the complexity of a structure except the normal 250-character line length.

The second type of instructions require two variables and perform the normal mathematical processes such as addition and multiplication, and other more complicated functions such as calculating polar angles from rectangular coordinates. They are used in the following way :

```
aplusb$=FNADD(a$,b$):REM aplusb=a+b
angle$=FNPOL(x$,y$):REM angle=ATN(y/x)
```

Since all the numbers are handled as normal strings then all the normal, standard Basic commands may be used, including PRINT, LEFT$,

Routines provided and their functions

Routine name	Function
num$=FNCONST("123")	|num$="123"
num$=FNNEG(num$)	|num$=-num$
num$=FNABS(num$)	|num$=ABS(num$)
num$=FNINT(num$)	|num$=INT(num$)
num$=FNSQR(num$)	|num$=SQR(num$)
num$=FNLOG(num$)	|num$=LOG(num$)
num$=FNLN(num$)	|num$=LN(num$)
num$=FNEXP(num$)	|num$=EXP(num$)
num$=FNSIN(num$)	|num$=SIN(num$)
num$=FNCOS(num$)	|num$=COS(num$)
num$=FNTAN(num$)	|num$=TAN(num$)
num$=FNASN(num$)	|num$=ASN(num$)
num$=FNACS(num$)	|num$=ACS(num$)
num$=FNATN(num$)	|num$=ATN(num$)
num$=FNADD(A$,B$)	|num$=A$+B$
num$=FNMUL(A$,B$)	|num$=A$*B$
num$=FNSUB(A$,B$)	|num$=A$-B$
num$=FNDIV(A$,B$)	|num$=A$/B$
num$=FNPOW(A$,B$)	|num$=A$ ^ B$
num$=FNMOD(A$,B$)	|num$=A$ MOD B$
num$=FNPOL(A$,B$)	|num$=ATN(B$/A$)
FNamDP(new)	|sets number of dec places
FNSUIT(num$)	|TRUE if Basic can express it

RIGHT$ and MID$.

Two other important facilities are provided by the routines, FNamDP and FNSUIT. FNamDP allows the user to set the number of decimal places to be returned (and a value between 0 and 18) by the routines. It is used in the following way :

```
past=FNamDP(14):REM Sets no. of decimal
  places to 14 and returns with past set to
  previous value.
```

The routine sets the default number of decimal places to 18. The second function FNSUIT is provided for convenience. Since all the numbers are stored in a string form it is perfectly acceptable to use the VAL instruction to determine the value of a string. However, because of the extended accuracy and range, some numbers will be unsuitable for conversion to Basic floating point variables and will result in a 'Number too large' error.

To allow you to determine if a number is suitable for conversion the FNSUIT command is provided. It will return a TRUE value if the number can be expressed by a Basic variable and FALSE if not. It is used in the following way :

```
IF FNSUIT(extnum$) THEN
  a=VAL(extnum$) ELSE a=0
```

Note the use of a floating point basic variable and not an integer one which would cause rounding to occur if the number has a fraction part.

When using the package it is best to use a mixture of extended precision numbers and normal Basic variables where possible, since the calculations with extended precision numbers are fairly slow due to the complex encoding and decoding that occurs with every command. It is advised that they only be used where strictly necessary (that is, when extended precision or range is required). Also, only use the minimum number of decimal places required since extra decimal places require extra processing and so slow the package unnecessarily.

```
10REM >$.Demo
20MODE12
30sides$=FNCONST("3")
40rad$=FNCONST("100")
50col$=FNCONST("3")
60CLS
70REPEAT
80GCOL VAL(col$)
90plus$=FNDIV(FNMUL(FNACS("-1"),"2"),sides$)
100xm$=FNADD("640",FNMUL(rad$,FNCOS("0")))
110ym$=FNADD("512",FNMUL(rad$,FNSIN("0")))
120PLOT4,VALxm$,VALym$
130angle$="0"
140WHILE VAL(angle$)<= 2*PI
150x$=FNADD("640",FNMUL(rad$,FNCOS(angle$)))
160y$=FNADD("512",FNMUL(rad$,FNSIN(angle$)))
170PLOT5,VALx$,VALy$
180angle$=FNADD(angle$,plus$)
190ENDWHILE
200FILL VAL(x$)-8,VALy$
210sides$=FNADD(sides$,"1")
220rad$=FNADD(rad$,"50")
```

CODE SEGMENT

```
230col$=FNADD(col$,"1")
240UNTIL VAL(sides$)>7
250PRINT'"Press SPACE"
260REPEAT UNTIL INKEY-99
270CLS
280PRINT'"Calculating factorial 123 ..."
290PRINT'"Note this calculation takes approx 50 seconds."
300x$=FNCONST("123")
310answer$=FNCONST("1")
320WHILE VAL(x$)>1
330answer$=FNMUL(answer$,x$)
340x$=FNSUB(x$,"1")
350ENDWHILE
360PRINT"123! is :";answer$
370PRINT'"e  is :";FNEXP("1")
380PRINT"PI is :";FNACS("-1")
390PRINT'"The square root of 99 is :";FNSQR("99")
400PRINT'"The polar angle of coords 4,6 is:";FNPOL("4","6");" rads"
410PRINT'"4 to the power 3456 is:";FNPOW("4","3456")
420PRINT''"(C) Copyright TJ Chappell July 1989"
430PRINT''"Please note that further help is provided by typing
LVAR at the prompt."
440END
>
```

```
Calculating factorial 123 ...

Note this calculation takes approx 50 seconds.
123! is :1.2146304367025324777E205

e  is :2.7182818284590845235
PI is :3.1415926535897932339

The square root of 99 is :9.949874371066199547

The polar angle of coords 4,6 is:9.827937232473298679E-1 rads

4 to the power 3456 is:5.239984827846554395E2888

(C) Copyright TJ Chappell July 1989

Please note that further help is provided by typing LVAR at the prompt.
```

```
Press SPACE
```