A FEW BUGS I HAVE FOUND

by Mike Robinson

SOLIDISK'S STLDISC PROGRAM

I've found rather a strange bug with the STLDISC program. I've been using it with a 32k SOLIDISK unit and though the size of the silicon disc is limited, I've found STLDISC very useful. However, one day a program I was developing kept crashing for no apparent reason and it took me ages to track down the source of the trouble.

In the end I found that a crash will occur if the drive is changed after doing a disc operation using STLDISC when the length of a program has the last two hexadecimal digits as 13, e.g. &13, &1A13, &2013, etc.

The problem can be recreated as follows:-
With STLDISC loaded enter the following one line program:

```
10REM STLDISC BUG
```

Check that the length of the program is exactly &13 (with PRINT ~TOP-PAGE). SAVE this to either proper disc or silicon disc. Then change drive!

Why this causes a crash I don't know. I wrote to Solidisk in March about this problem but have not had a reply.

EXTENDING A FILE WITH WATFORD 1.3 DFS

This problem occurs when I have OPENUPed (with Basic II) a file for reading and writing and use BPUT to write additional data to the end of an existing file. This works fine and the file will be extended to include the new bytes, providing BGET is not used on another part of the file before CLOSEing it. The extra bytes will not be on the end when you next re-open the file.

I'm surprised I've not seen any reference to this problem in any of the computer press as I would have thought it was a fairly common situation. I have personally come up against it on three programs involving random access files which I've written recently. Incidently the problem does not occur with the Acorn DFS.

I wrote to Watford Electronics about this and they replied to say my letter had been passed to the author of the DFS to check that it had been fixed. Some work had been done to the text filing system and they believe that the problem had been corrected. Other improvements of the latest version are full tube compatibility and a slightly faster disk format. There would be a £5 charge to upgrade to the new version.

BASIC'S 'PRINT' FORMAT

The use of a semi-colon after the PRINT statement is pretty elementary. If a semi-colon is used before any numeric values, they will be printed at the cursor position without any leading spaces. But if no semi-colon is used, numbers will be 'formatted' to the right side of the current field width. This will be 10 spaces unless the @% variable has been altered (see User Guide).

e.g. PRINT 123 will produce:

```
    123   (with seven leading spaces
```

whilst PRINT;123 will produce:

```
123           (no leading spaces)
```

Now the problem I have found is if you want numbers formatted both ways in the same PRINT statement. As soon as the semi-colon has been used, all following numbers will be treated as if the semi-colon has been used even without it. So the statement PRINT "A=" 123 " B=";456 " C=" 789 will produce:

```
A=       123 B=456 C=789
```

and not as you would expect:

```
A=       123 B=456 C=       789
```