WIZZDOM

A Newsletter for the owners of Dick Smith WIZZARD and FUNVISION computers

Issue 4- written by Barry Klein

EDITORIAL

It seems that Dick Smith Electronics has finally abandoned the WIZZARD, with the possible exception of one last shipment of hardware and cartridges. DSE gives no support to us, as a User Group; I only find this information by asking store staff or getting information from other members who have got different information from other store staff. Because of this, I believe that we, as the WIZZARD User Group are out on our own when it comes to support for the computer i.e. any new developments will come from us, not from Dick Smith Electronics.

DAMNATION

Another error has been found in Issue 1:

Page 5, line 2. the values for locations 512 to 515 should be: 8,248,63,255.

Some Thoughts on Programming

You'll find that nearly all books on programming describe FLOWCHARTS and will virtually demand that you draw one before you start to write (code) your program. However, if you asked programmers whether they draw flowcharts before they start to write, they'll usually say that they don't. However, you can be sure that these people have virtually drawn a flowchart 'in their head' before starting. Flowcharting 'in your head' is OK for simple programs (but see the 'YES/NO Trap' later), but is not good for complex ones: you'11 invariably remember some vital point when you're well into your program, usually requiring a major revision. Especially for the more complex programs, the FLOWCHART should be part of the program's DOCUMENTATION.

An important part of documentation is the comment in the program itself. In BASIC, this is the REMark statement. Often, these 'internal comments' are the only documentation that exists for a program. It is most important when the program needs modification months or years after being written. Without comments, even you may have forgotten what the program does and how it does it. You will then spend a lot of time analysing the program and critcising yourself for not putting in comments originally. A problem with some microcomputers (the WIZZARD is one) is that memory limitations may permit the use of only a few REMark statements. In this case, you should write separate documentation and keep it with a 'LISTing' of the program.

If, while you are writing your program, you run out of memory, you could continue to program after deleting REM statements (and, with the WIZZARD, after CSAVE and CLOAD). If you have not used GOTO and GOSUB statements to REM statements, then you do not need any further changes. If not, then you'll have to edit all those GOTO and GOSUBs. Another reason for not using REMs as the destinations of these statements is that they still take some time for the computer to decide that they are to be ignored. Your program will be faster to run and easier to modify if you don't GOTO or GOSUB any REM statements.

The YES/NO trap.

When drawing your flow chart, you will use the 'decision diamond' when you test for some condition. This is usually indicated by a question in the diamond and two alternative routes away from it - 'YES' and 'NO'.
This can lead to problems if you don't take sufficient care with the question as the following example shows.

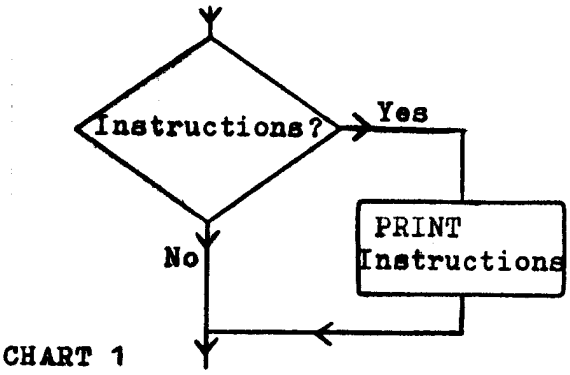

CHART 1

This seems innocent, but it is ambiguous. The reason is simple: the possible answers to the question are not just 'YES' and 'NO', but any combination of keyboard characters, including none at all.
When you convert this flowchart into a program, you can do it in two ways:

```
110 PRINT"DO YOU WANT INSTRUCTIONS";
120 INPUT A$
130 IF A$<>"YES" THEN 510        or       130 IF A$="NO" THEN 510
210 etc PRINT instructions
510 start program
```

In the first case, any answer other than "YES" will not print the instructions, while in the second case any answer other than "NO" will print the instructions.
The correct way to approach the decision diamond is to ask a question whose answer can only be 'true' or 'false' and to use those answers instead of 'yes' and 'no'. A better flowchart and its program are shown on the next page.

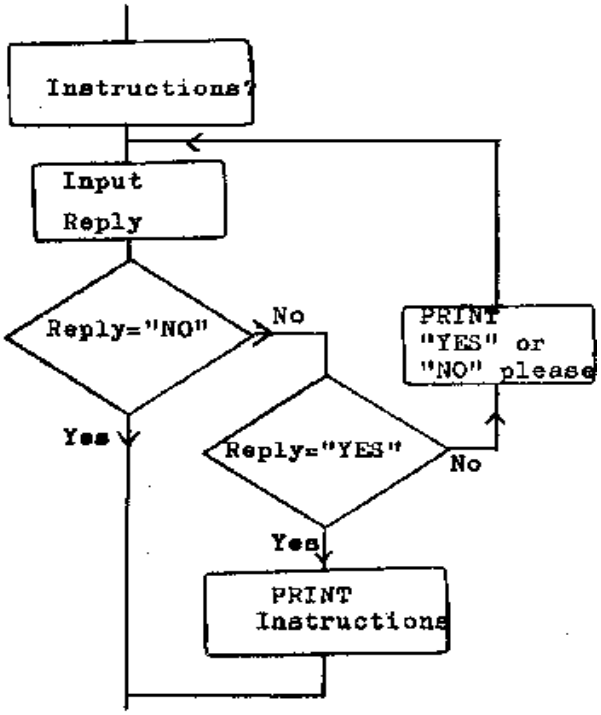

CHART 2

```
110 PRINT"DO YOU WANT INSTRUCTIONS";
120 INPUT A$
130 IF A$="NO" THEN 510
140 IF A$="YES" THEN 210
150 PRINT"'YES'" OR 'NO' PLEASE"
160 GOTO 120
210 etc PRINT instructions
510 start program
```

N.B. You can allow for 'Y' or 'N' to be valid replies with:

```
125 A$=LEFT$(A$,1)
130 IF A$="N" ...
140 IF AS="Y" ...
```

This will also allow answers such as 'YECCH' or 'NATURALLY', but this won't be a problem with sensible people using the program.

The following examples, taken from the two books of WIZZARD programs, will illustrate some of these things.

The 'First Book' was written wholly by one person, so the style is consistent throughout. The programs are generally well designed and have enough comments to explain what is going on. There is, however, one glaring omission and one not obvious omission. The former is that there is no explanation of the 'I=RND(1)' or similar staements anywhere. The latter is that all 'COLOR' statements follow 'CLS' statements, never the reverse. Reasons for this should have been given, at least in an introduction to the book. (The reasons are given in WIZZDOM Issue 2).

The 'Second Book'was written by many people, probably all "amateur" programmers. The programs are often poorly structured, use poor programming techniques and some have no comments at all. As each person wrote his program for his own version of BASIC and was unaware of the existence of any other version, most of these programs will not operate proper1y on the other version, without change.

Example 1. WIZZARD Number Guesser, First Book, p20

There should be a comment for line 50 near line 50, as well as the one at line 8000. Line 50 should be 'GOSUB 8010'.

Line 60 should be explained (see above)

You can only stop the game with RESET or CONTROL C, and the end of game message is not very clear: it should be 'MOVE JOYSTICK FOR NEW GAME'. It would have been even better to ask 'DO YOU WANT TO TRY AGAIN' and look for a 'YES' or 'NO' as an answer.

Example 2. Copter Jump, Second Book, p57
This program has no comments (REM statements) at all. They are needed, especially for lines 3115 to 3154. It is likely that the blanks between the quotes in the PRINT statements should really be 'control' characters which will display a picture. Note the CHAR statements for characters number 2,3,4,6 and 1 (lines 3010 to 3071). These characters are not used in PLOT statements and characters 14 and 15 are redefined before they are. These characters correspond to CONTROL B, C, D, F, A, N and O respectively.
Lines 130 to 134 show that the program was written using 1982 BASIC, so it won't run correctly with 1983 BASIC.
A good point is the controlled way of stopping or rerunning the program - lines 9305 to 9330, although the expected values should have been explained.

PRESTIDIGITATIONS

1. Printers and the WIZZARD

The WIZZARD's 'Parallel I/0 Interface' (usually called the 'printer interface') has a CENTRONICS type of printer port. The supplied cable connects leads 1 to 10 (data and control) and pin 16 (common/earth) on the CENTRONICS plug. These connections will work with most popular printers designed to work with microcomputers. It has been tried with EPSON printers, the TOYO TP-40, a BX-80, Seikosha GP 100 and a C-Itoh 8300p.
The WIZZARD has two limitations when using LPRINT:

a. only 128 characters can be printed. CHR$(129) is the same as CHR$(1).
b. each LPRINT (and LLIST) command is preceded by the DEL character (CHR$(127)) and succeeded by the NUL character (CHR$(O)). A NUL character in a PRINT or LPRINT statement will cause the statement to terminate e.g. PRINT"ABCD";CHR$(0);"EFGH" has the same effect as PRINT"ABCD"; . NOTE: the DEL character at the start of the LPRINT statement will not affect most printers, but causes the TOYO TP-40 to print a character (☒) and prevents it from being set into graphics mode. However once graphics mode has been set (see below), the DEL character is ignored.

The following statement will put the TP-40 into graphics mode:

```
LPRINT CHR$(8);CHR$(18)
```

CHR$(8) is 'backspace': this serves two purposes - it backspaces the printhead over the printed DEL character and it allows the CHR$(18) to be recognised.
The program on the next page can be used to print any of the 256 possible characters. It does not add characters that are not wanted. It will operate more slowly than an LPRINT statement of course.

```
100 REM INITIALISE PRINTER
110 GOSUB 9710
120 REM PRINT ALL PRINTABLE CHARACTERS
130 FOR A=32 TO 127 STEP 32
140 FOR B=O TO 31
150 C=A+B
160 GOSUB 9810
170 NEXT B
180 LPRINT
190 NEXT A
200 LPRINT
210 FOR A=160 TO 223 STEP 32
220 FOR B=O TO 31
230 C=A+B
240 GOSUB  9810
250 NEXT B
260 LPRINT
270 NEXT A
280 STOP
9700 REM ROUTINES TO PRINT 'ANY' CHARACTER
9705 REM INITIALISE PRINTER
9710 POKE 59393,0
9720 POKE 59392,255
9730 POKE 59393,52
9740 RETURN
9750 REM


9800 REM PRINT CHARACTER (ASC VALUE IN 'C')
9805 REM CHECK PRINTER NOT BUSY & CLEAR FLAG
9810 IF PEEK(59393)<128 THEN 9810
9820 Q=PEEK(59392)
9830 REM PRINT CHARACTER
9840 POKE 59392,C
9850 REM 'STROBE' CHARACTER INTO PRINTER
9860 POKE 59393,54
9870 POKE 59393,62
9880 RETURN
9890 REM WITH 'FAST' PRINTERS (NOT TP-40)
9900 REM LINE 9810 MAY NOT BE NEEDED.
9910 REM INITIALISE ONLY NEEDED ONCE,
9920 REM NOT NEEDED IF ALREADY USED
9930 REM LLIST OR LPRINT.
9940 END
```

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
 。「」、・ヲァィゥェォャュョッーアイウエオカキクケコサシスセソ
タチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワン゛゜
```

Output from program

LISTING 1.

This program is specifically for the TP-40 which doesn't print every character from 128 up. For other printers, you can change line 210 appropriately. E.g. 'FOR A=128 TO 255 STEP 32'. To prevent the DEL character being printed by the PRINT statements, replace them with:

```
C=13                (carriage return)
GOSUB 9810
C=10                (line feed)
GOSUB 9810
```

PRESTIDIGITATIONS (cont)

More on SELF MODIFYING PROGRAMS (cont. from Issue 3)

It is possible to accept characters from the keyboard and write them into the program area so that CSAVEing the modified program will save those characters. This is a more useful type of program than the one given in Issue 3, however WIZZARD BASIC seems to have been written in such a way as to make the job as difficult as possible. There are many restrictions which prevent this type of program from being as useful as it could be.

The program can write data into itself in two ways: with PRINT or DATA statements. The DATA statement writes the characters exactly as they appear (except for spaces), then adds a carriage return (CHR$(13)). With the PRINT statement, however, the expressions in the statement are evaluated, converted to characters, then 160 is added to each character before writing to memory. No extra characters are written, but you have to be careful about the values used in locations 218 and 219 or you'll write into the wrong area of memory. Because the PRINT statement adds 160 to the character, it can be used to write control characters into the program. Characters 96 to 127, when PRINTed, become characters 0 to 31. NOTE that the only way of writing a space into the program is with DATA" " - you then have the problem of eliminating the quotes characters as well as the carriage return.

WIZZARD USER GROUP PROGRAM LIBRARY

Thank you to all those who have sent programs for the library. However, I would like to state my policy for programs to be distributed through the library: I will not distribute programs that I know are copyright (e.g. those sold by Dick Smith or published in his books) or that I suspect may be copyright (e.g. they look as though they're taken from a book of games). I also reserve the right not to include a program in the library if I feel that it is not particularly interesting. When possible, I will send some comments to the authors of 'failed' programs to try to help them.

The following programs have been added to the library.

HI-RES AUTHOUR I and II

These programs are designed to help you produce graphic characters (I) and multi-character shapes (II).

COPTER PILOT

Pick up aid for the poor countries of Africa (graphics designed using HI-RES AUTHOR. All 3 written by P. Pascoe)

DAVE'S SKETCH

Yet another sketch program.