Odds And Ends

String Array Bug

J. Horemans

If you use string arrays for example A$(N), then be aware of OSI's program bug that can wipe out your program if the FRE function is called by the machine when leftover strings fill your RAM. Avoiding the problem is simple. Merely choose the DIMensions of the string array with the formula N = 3* + 3. For example DIMA$(44) is O.K., but DIMA$(45) can cause a program to crash. Also call the FRE function after doing a string array manipulation. Write yourself a program to find (and then make a permanent list of) the values that circumvent this problem. Here is a suitable example.

```
10 for I = 1 to 20 (or any other values e.g. 21 to 40)
20 N = 3*I + 2
30?N
40 next I
```

You should get a list like:

2, 5, 8, 11,14, ... 44, 47, 50, ...

The values of N printed on the screen are the ones to use. Choose the one closest to the size array you need. It is also necessary to put a line like this after string array manipulations:

```
100 X = FRE(0)
```

For Those Inclined To Experiment

Here is a program to demonstrate the string array bug in OSI.

```
 5 A$ = "B"
10 FOR I = 1 TO 100
20 B$(1) = B$(1) + A$
30 next
40 B$(1) = ""
50 X = FRE(0)
60 A = A + 1:?"DONE"A;X
70 GOTO 10
```

Run the program as is. Nothing happens for a few moments, then the screen starts to flicker. This is the external symptom of the string bug. Recover by pressing the break key and doing a warm start by pressing W. Now add line one:

```
1 DIMB$(2) (or use any number generated by 3*N + 2)
```

The computer should now hum through this little program, telling you it is happily doing its job and, in the process, cleaning up the leftover string in line 50. Remember, it *is necessary to have a line like line 50* to call this function. Try running the program without line 50. If the DIM statement is o.k. the screen won't flicker, but my 8K machine never gets past DONE 1.

If you are making a graphic display, and the o.k. message keeps coming up to mess it up, here are two ways to defeat it.

1. In immediate mode (no line number):

```
POKE 3, 96
```

This will turn off the message altogether, but you have to press RETURN to get the cursor back to the left of the screen.

2. Just before your subroutines, put this line:

```
3990 FOR I = 1 to 10000: NEXT:END
```

Your program will now wait for the time specified by counting from 1 to 10000 (or whatever number you put in) before flashing the o.k. message and ruining your display. If your subroutines aren't at the end or you haven't any yet, just use any convenient high number for the line.

Those fast screen clears are great, but those pokes to locations 11 and 12 can be hard to keep in order if several ML calls are needed. Here is a fast full screen clear that does NOT use the ML call via USR(X). The method is described in *The First Book of OSI* by Williams and Dorner. I have adapted it to the C1P/Superboard screen. It does take more memory than some machine screen clears, but many of them need DATA lines to POKE in too, and that can cause confusion with your DATA statements.

```
5000 A = PEEK(129): B = PEEK(130):POKE129,0:
   POKE130,212
5010 S$ = "                ": rem 31 BLANK SPACES
5020 FOR I = 1 to 32:S$ = S$ + " ":NEXT:POKE 129,A:
   POKE 130,B:RETURN
```

To call this screen clear insert a line like this:

```
100 GOSUB 5000
```

©