**Program Step - A Fast, Flexible Debugging Aid**

Modern languages have released developers from a lot of the shackles associated with the first and second generation programming languages, but some really useful things have fallen by the way-side in the process.

Not many will mourn the passing of mandatory line numbers in code but it is indisputable that they could make life really easy for tracing bugs. MMBasic does maintain the popular TRON/TROFF (although its future is not certain) but TRON is indiscriminate and usually involves screen capture and then going through the lines with your editor to even identify the section of code - it isn't a nice experience relating mountains of 100 101 373 374… etc. to your nicely formatted code in the editor and the numbers will change as you add/remove lines each run. Another option might be to use an IDE which can be very detailed but often a massive over-kill for simply tracing an annoying bug - have you played with the test harness in .NET? ugh!

Below is a method for a tracing under control of the programmer/program at run time. You simply turn debugging on at the point you require and add “PSteps” at key points, no longer do you get a trace for every line (even blanks). You only get output when you want it, and the PStep will be the same each time. Familiarity with the numbers becomes meaningful whilst debugging, whereas with TRON it is largely meaningless and you end up starting from scratch each time you debug your code.

A simple flag is switched to determine if debugging is required or not - how you do this is down to you but it should be simple to make it fast. Next a SUB should be defined to provide the debugger output in whatever format you like. Again, the SUB should be simple and optimized for speed.

The SUB can now be called at points in the program which provides crucial information for the point in the code currently executing - when that error occurs, you have pin-pointed the sections of code involved with the level of detail you want.

The following provides micro-control of flexible debugging tell-tales and it's small and fast enough that you can simply leave it in your code when you are done

```
  Dim debug as Integer' seems a waste for a simple flag but it's faster
  debug=1 ' switch debugging on

  timer=0
  Pstep 1000' call the debugger with a unique  program step identifier

  For n=1 to 10
     PStep 2000
  Next j ' deliberate error, take out the j to test success

  Pstep 3000

  Print timer ' how long did that take?

  Sub PStep(x As Integer) ' the debugger routine
    If debug Then Print "PStep ";x ' optimized for speed - change this bit
to do whatever you want as part of the
```

```
                                        ' tell-tale procedure but it might get
called a lot so keep that in mind
  End Sub
```

The debugger is called each time but if the debug flag is off, it rapidly exits without doing anything - the overhead is quite small. You might want to output Strings instead of numbers which will add a really nice dimension to your debugging but beware; string handling is time consuming and will eat through your program space quite quickly so keep it pared back to the necessary minimum.

Now you can switch debugging on and off at the relevant points in your code and use steps that mean something to you - try to make the numbers used “structured” so, 10000+ are functions, 20000+ are Subs and anything below is main program etc…

The code is light and can be left in when your project is finished with only a tiny overhead. If you are pushed for program space change the names to something shorter to save a few bytes per call