Five Debugging Tips

(intro)

Tip 1: Most people think of Debug and Trace as their only debugging windows. Not true. The View window is invaluable in tracking down erroneous behavior.

You're able to determine which tables are open, which table is in the current work area, and whether or not an index is set, how many records are in that table, and what relations are set.

This is a lot of valuable information in a compact space. I've often solved (or avoided) a problem by noticing that a certain table isn't open, or that the table in the current work area doesn't have an active index.

Tip 2: Run in 50 line mode in DOS or an enhanced resolution in Windows. In conjunction with Tip 1 above, I position my windows as follows:

View window in the lower right corner, with the right third of the window off the screen (there's generally not anything valuable in there anyways).

Command window, sized approximately 40 columns by 8 rows, above the View window, so that if I have to run a command during Suspend, the Command Window isn't hidden behind the application.

Debug goes to the left of View, but leaving the first 10 to 15 columns on the screen uncovered, so that any text echoed to the desktop is immediately visible.

Finally, Trace goes above Debug. The sizes of Debug and Trace vary according to how heavy I'm using them.

```
----------------------------------------------------------
|                                                        |
|                                                        |
|                                                        |
|                                                        |
|                                                        |
|                (application window here)               |
|                                                        |
|                                                        |
|                                                        |
|                                                        |
|                                                        |
|                                                        |
|........................................................|
|        _____________________|                          |
|        |                    |                          |
|        |                    |      (command)           |
|        |                    |                          |
|        |    (trace)         |                          |
|        |                    |--------------------------|
|        |                    |                          |
|        ---------------------|                          |
|        |                 |                             |
|        |                 |                             |
|        |                 |        (view)               |
|        |                 |                             |
|        |                 |                             |
|ON      |    (debug)      |                             |
|1005    |                 |                             |
|Master i|                 |                             |
|OFF     |                 |                             |
----------------------------------------------------------

  ^
  |___ Commands that echo to the screen can still be seen here.
```

Tip 3: Memory variables are not the only objects that can be placed in the Debug window for evaluation. You can place any FoxPro function or expression in Debug. Some obvious expressions include RECNO(), BOF(), EOF(), RDLEVEL(), RECCOUNT(), and so on.

Some of my favorites are SYS(16) (determines which program or function is running), VARREAD() (the value of the current object), and a number of SET commands. For instance, if a SEEK is failing, you might place SET("NEAR") and SET("EXACT") in Debug to determine if one of these is changing.

Furthermore, you can place expressions in Debug. The expression RDLEVEL() > 2 will show .F. until you've hit a third read level in your system.

Tip 4: Use the break point functionality of the Debug window. Clicking on the middle bar of the Debug window next to an expression will display a diamond (DOS) or a shaded ball (Windows). When the value of this expression changes, your application will be suspended.

How many times have you run into "End of file encountered" without expecting it? Setting a break point on EOF() will stop your application the instant a command is issued that positions the record pointer at the end - such as a COUNT.

Try using the break point on a logical expression. In the RDLEVEL example above, you may not care what the actual read level is, but you do want to know when it goes above 3. When that fourth RDLEVEL is encountered, your system will be suspended and you can more accurately pinpoint the reason.

Tip 5: Now that you've gotten the hang of Debug, you'll find your Debug window filled with expressions. However, when you switch to another system, you'll have to get rid of each application-specific expression individually - and this is a nuisance.

Many programmers know that you can use the Ctrl-End key combination to clear out all expressions at the same time. Unfortunately, this also closes the Debug window, which is also a nuisance.

I've setup a keyboard macro as part of my default set that will switch to the Debug window, issue the Ctrl-End combination, and then - this is the good part - open up the Debug window again. The entire macro looks like this:

```
{F10}wd{CTRL+END}{F10}wd
```

Using this macro several times a day will probably save me 10 or 15 minutes this year - which I can easily waste by making another dumb mistake requring another go at the system with Debug.

(Whil Hentzen is....)

<EOF>