[7.39] Quickly delete locked, archived variables

Two steps are required to delete an archived variable: first you unarchive the variable, then delete it. If the variable is also locked, three steps are required. These steps are necessary whether you use the functions *unlock()* and *unarchiv()*, or use the VAR-LINK screen. This simple program deletes any variable or list of variables, regardless of whether they are archived or locked.

```
delvar1(n)
Prgm
©("v"), ({"v",...}) Delete v locked/archived
©26decØ1/dburkett@infinet.com
local k                              © Loop counter

If getType(n)="STR"                  © Convert single argument to list
{n}→n

If getType(n)="LIST" then            © Delete list of variables
for k,1,dim(n)
  if getType(#(n[k]))≠"NONE" then    © Ignore nonexistent variables
   try:unarchiv #(n[k]):else:endtry  © Must unarchive before unlocking
   try:unlock #(n[k]):else:endtry
   delvar #(n[k])
  endif
 endfor
else                                 © Display argument type error
dialog
  title "DELVAR1"
  text "Bad argument type"
 endDlog
endif

EndPrgm
```

The variable name argument is a string or list of strings. To delete a single variable *var*, use

```
delvar1("var")
```

To delete more than one variable, use a list:

```
delvar1({"var1","var2","var3"})
```

A single string argument is first converted to a list, so either argument type is handled consistently, which saves some program memory. Variables which do not exist are ignored. This may not be the most appropriate behavior, but at least it is consistent with the behavior of *unarchiv()*, *unlock()* and *delvar()*. An error message is displayed in a dialog box if the argument is neither a string nor a list.

This program deletes variables which are locked, archived or both. The order of the *unarchiv* and *unlock* commands is important, because a variable must be unarchived before it is unlocked, or a "Memory" error occurs. This program works because no errors occur if you unlock or unarchive a variable which is already unlocked or unarchived. This means that the program can also be used as a quick way to delete a list of variables which are not locked or archived.

The *unarchiv* and *unlock* commands are used in *Try ... EndTry* blocks, otherwise an error occurs if you try to delete system variables. Many system variables can be deleted, and it is often useful to do so.

This program should obviously be used with some care, since there are no prompts to verify that you really want to delete the variables.