Mastering Macros
by John Rethorst

Going Global

Up to now, we've used read-write variables that we wrote a value to, and then could read that value from, while the macro was running. When the macro ended, the values simply disappeared. We call these *local variables*, and they're nice for many uses. At other times, though, we need a variable that will hold its value after a macro ends – until you quit WP or write something else to the variable. These are *global variables*. Like the local variables, there are 50 of them, denoted **GlobalVar00 - GlobalVar49**. Let's see what we can do with these.

Making a glossary

One use is to keep any piece of text until you need it. Put access to the contents of several global variables on a menu, and you have a glossary feature. I wrote one of these, with a menu giving you up to 26 glossary entries, each up to 255 characters (the maximum string length a WP variable can hold).

Aside from the length limit, there's quite an advantage to putting a glossary entry in a variable rather than putting the actual text in the macro script, as we did last month with an address on a work menu. Simply, the entry is much easier to change if we put it in some reference document, which a macro could read, and then put in a variable. You could of course change the text in the script itself, but as you become more advanced with macros you'll find yourself writing them for associates as well as yourself and, if your associate isn't learning macros too, asking him/her to edit a script is a daunting task. So let's do it better.

For a start, create a new document with a table of one column and 26 rows. Don't have any blank lines above the table. Name this document "Glossary File" and save it in the WordPerfect folder in the Preferences folder in your System Folder (or leave the Preferences folder out of that path if you're using system 6). Put a few words of text in the first cell of that table, and close the file.

Now we want to cook up a macro that will open this file (probably when you start WP), and put the contents of that cell into a global variable. A second macro, which we'll call when we want to insert the glossary entry, will put the contents of that variable on a menu, and the user can just click on it. The macro will then type the variable into the document at the insertion point.

The macro to assign entries

So the first macro starts by opening the glossary file you just made. How does the macro find it? When you start WP, one of the things it does is ask your Mac for the location of its system folder, and then puts that into a read-only variable called **BootDir**. So we can start the path with that, and not worry about what you or anyone else has named their hard disk, or how many levels deep the System Folder is, and so on. The first command is then:

```
Open Document (BootDir$"Preferences:WordPerfect:Glossary File")
```

with the join operator ($) hooking up the read-only variable with the character expression containing the rest of the path to the file. As a character expression, it's in quotes.

With the file open, and a table at the top of the file, the insertion point will be in the first cell. From there we can :

select the contents of the cell
copy
assign a global variable to the clipboard

and, since people generally start using variables 1, 2, 3 etc. in their macros, let's start using globals from the other end, the better to stay out of another macro's way. Since our globals are going to stay around as long as WP is running, the other guy's globals will too. So we'll start with GlobalVar49. Make a new macro entitled "Assign Glossary Entries" (no need for a keystroke), and put this script in it:

```
Select TableCell
Copy
Assign (GlobalVar49;Clipboard)
Close
```

We'll add to this later, but all we'll need to add are repetitions for more glossary entries, to be contained in more globals.

A menu for global variables

Our next macro will put the globals (the one we have now, and more later on a menu. A case command will follow, to send macro execution to the proper label. Make another macro, call it "Glossary," and assign it a keystroke you like. Start it off with this script:

```
Menu (Var01;"Glossary";{GlobalVar49})
Case (Var01;{1;49};cancel)
Label (cancel)
End Macro
Label (49)
Type Var (GlobalVar49)
```

and let's look at a few points. First, we're using a local variable, Var01, for the menu and case commands. Why not? We can use both local and global variables within one macro. We'll have no need for the contents of Var01 after this macro ends, so a local variable is a good place for it.

Second, there's a new command in the script above, and an important one: **Type Var** will put the contents of any variable into the active document at the insertion point. Not a bad macro command for a word processor to have.

Third, we've written menus with *character expressions* (i.e. regular text), as in:

Menu (Var01;"Menu Title";{"Menu Choice 1";"Menu Choice 2"})

where text has to be in quotes. The Glossary menu above uses GlobalVar49 as a menu choice, and it's not in quotes. If it were, you'd see, literally, "GlobalVar49" on the menu – probably not much help. But the *contents* of a variable can be used in place of a character expression, where the variable replaces the expression and the quotes marking it as such. Think of the variable containing text the same way that quotes contain text.

The point of doing the menu this way is greatly expanded WYSIWYG. This menu won't say something like "Glossary Entry 1" but will show the actual text written to the variable, or as much of it as will fit on the menu.

Try these macros out: run the Assign macro first, then the Glossary macro. You see the potential, with more entries. You also see the potential for error, if the user runs Glossary without running Assign first. One way around that, for the users who need a glossary feature on a regular basis, is to have their **OnStartup** macro run the Assign macro. OnStartup runs whenever you start WP (and **OnOpenDocument** runs whenever you open a document containing it or, if it's in the Library, whenever you open any document), so a line in OnStartup could be:

```
Run ("Assign Glossary Entries")
```

which is one way to run one macro from another.

But our user, bless his little heart, may not want to add that line to OnStartup. Well, in any case we can have the Glossary macro check to see if Assign has been run – that's to say, see whether it's put anything in GlobalVar49. Put these lines at the top of Glossary:

```
If (GlobalVar49="")
Run ("Assign Glossary Entries")
End If
```

and the structure of things is largely finished. All we need to do now is add the code for multiple entries. As you might expect, this will be fairly repetitive, as is a lot of code. Copy and Paste are good friends when writing a comprehensive script.

For the Assign macro, though, we need to tell the macro how to go from one glossary entry (that is, one cell in the table) to the next. What command would we use? **Down ()**? Nope. It moves the insertion point down a line, which may be in the same cell for a multi-line glossary entry. Here's a good place to click **Pause** at the top of the script editor window, and go back into the program to see what it does. Play with a table for a minute and you'll see that **Tab** is the best keystroke to go from one table cell to the next.

As a nice touch for the user, you can add a **prompt** while assigning entries. This line, at the top of the Assign macro, would be something like:

```
Prompt (75;125;"Glossary";"Reading glossary information . . .")
```

where the first and second parameters are the distance in pixels/points from the top left of the screen to the top left of the prompt. The third parameter is the title, and the fourth is the text.

Put an **End Prompt** command at the end of the Assign macro, so as not to leave the prompt on screen.

You should be able to figure things out from here but, if you run into trouble, the complete macro set accompanies this issue of the News.

Conceptual point: in past columns I've talked about recording part of a macro and scripting the rest. Here, though, we've scripted part of things and then gone back to the program not to record but just to see how things work. We then take that observation and write our script to fit.

Quiz time

Answer to last month's quiz: the code to print the current page is:

```
Print Options (PhysicalPage;PhysicalPage;1;Document;Every Pa
ge;Forward;Print Overlay)
Print (Document)
```

This month's quiz: the user may have other data in some of these global variables, and may have forgotten that. If he or she then calls Glossary and if GlobalVar49 is empty, Glossary will run Assign Glossary Entries and overwrite anything else in globals 24 through 49. As an alternative, rewrite the Assign macro so that if GlobalVar49 is empty, the macro posts an Alert saying that the Assign macro needs to be run.

* * *

John Rethorst, author of *Teach Yourself WordPerfect*, finds macros mellifluous, mystical and majestic.