COOL TOOL OF THE MONTH: GENMENUX

By Whil Hentzen

[Sidebar text]

Take control of your menu generation process with GENMENUX, a wrap-around shell for FoxPro's GENMENU.

[Article]

Unless you've been living in the traditional cave for the past year, you've heard about Ken Levy's GENSCRNX, the pre- and post-processor for FoxPro's GENSCRN. Not a replacement, GENSCRNX takes the SCX/SCT files, massages them, runs them through GENSCRN, and the massages the output before handing you a final SPR.

There is a corresponding processor for FoxPro's menu generator that hasn't received as much airplay. Steven Black originally created GENMENUX as a companion to GENSCRNX in order to provide automatic menu translation for his INTL GENSCRNX Driver. Andrew Ross MacNeill took the baton and made additional modifications to make GENMENUX a more powerful and generic tool.

Probably more than any other question about these tools, the one I get asked is "So I've heard all about these tools. What can I actually do with it?" This article will get you up and running with GENMENUX before the next commercial.

The Problem

You don't have much flexibility when dealing with the FoxPro menu builder, and this has frustrated some developers to the point of manually modifying the MPR. For example, FoxPro does not provide you with the capability to remove shadows or margins or change the colors of bars and pads from menus.

In addition, many developers find themselves repetitively creating menus with the same objects, going back and adding hot keys that they forgot the first time through, and resorting to a number of other kludges.

The Solution

The "X" series of pre- and post-processors provide additional functionality to FoxPro's Power Tools. In particular, GENMENUX gives you, the developer, the ability to:

Control the appearance of menu bars and pads - including shadows, margins, and colors.

Control the positioning of menus and popups.

Add messages to DOS menus and override the Windows Message clause.

Completely remove (not disable as SKIP FOR does) one or more menu bars based on logical conditions.

Create common groups of objects and place them in libraries that can be inserted into menus with a single call instead of repetitively creating them.

Call programs ("Drivers") to do additional processing during the menu generation process.

Override the GENMENUX directives on a selective basis.

How GENMENUX works

Both GENSCRNX and GENMENUX use the same methodology (I'll refer to GENMENUX to make the wording easier). First the original tables (MNX/MNT) are copied, and modifications are made to those copies. Next, FoxPro's GENMENU is run against the modified MNX/MNT. Finally, GENMENUX takes the resultant MPR and performs final processing.

Note that the original tables are never modified themself. This means that you can make repeated modifications through the Menu Builder as intended - the MNX/MNT is then shot through GENMENUX again.

Drivers - what are they and why?

This is a good time to explain the concept of drivers. Obviously, GENMENUX.PRG is actually a series of programs, each performing a specific function. Between each of these functions, GENMENUX looks for an external program and if it finds it, runs that program before continuing on to the next section.

For example, you can place the directive

```
*:MNXDRV1 my_prog
```

in the Setup comment snippet. Then, after the copy of the MNX/MNT file is created and opened, MY_PROG will be run. Thus, you could make your own modifications to the MNX/MNT files before going on to the next step in GENMENUX.

Obviously, creating a driver isn't for the faint of heart, and requires close attention to detail (as if the rest of programming doesn't). When each of these drivers is called is listed in the documention.

How to install and run GENMENUX

Now let's get down to our GENMENUX jump start. We're going to suppose that you have a menu that you've created and is ready for modifications.

1. Place GENMENUX.PRG in your FoxPro directory.

2. Place the following line in your CONFIG.FP file

```
_GENMENU = G:\FPD\GENMENUX.PRG
```

modifying the path as needed to suit your environment. Then start FoxPro up again.

3. Open up your menu and generate it again. You can do this manually (through the Program, Generate command, or via the Project Manager. You'll see a second thermometer bar indicating that GENMENUX is doing it's thing.

4. Examine the resulting file - IT.MPR. You'll see a notice in the SEETUP that indicates...

```
** Menu Builder Enhancements by GENMENUX 1.1a  **

** This file has been modified using
**   GENMENUX 1.1a  - FoxPro Menu Processor **
```

What to do with GENMENUX

The big question on everyone's mind is "So I've heard all about these tools. What can I actually do with it?" Here are four types of functionality that you can use immediately:

1a. FoxPro for Windows has a Message clause that displays a message for each menu bar as the cursor scrolls from bar to bar. To provide this same functionality in DOS, you had to resort to a kludge.

To displays the message "Earth to Whil" for a menu bar, place the text

```
*:MESSAGE "Earth to Whil"
```

in the comment snippet.

1b. To display the contents of the memory variable m.cLastMessage for a menu bar, place the text

```
*:MESSAGE cLastMessage
```

in the comment snippet instead.

2. Normally, FoxPro adds a space to each side of the menu bar. To get rid of this space, place the text

```
*:NOMARGIN
```

in the comment snippet.

3. To turn off the shadow that accompanies a menu pop, place the text

```
*:NOSHADOW
```

in the comment snippet.

4. This is my favorite. As you know, you can selectively disable menu bars by placing expressions in the SKIP FOR clause. However, the bar is still visible, and I don't like this. I prefer to make the whole bar disappear, and this can be done, again, controlled by an expression, with the IF directive. Here are three examples.

4a. Suppose your menu will be run under DOS and Windows, and you have a menu option for a color picker that you want displayed only if you're in DOS. Place this text in the comment snippet:

```
*:IF _DOS
```

Let's examine the MPR to see what GENMENUX does with this. In the Cleanup Code, you'll find the construct

```
IF NOT _DOS
   RELEASE BAR 9 OF PREFERENCE
ENDIF
```

Note that GENMENUX took the expression in the snippet and preceeded it with a NOT, and then issued the RELEASE command. In other words, the menu bar will be displayed if the condition is true.

4b. You could also selectively display a menu bar depending on the version running. The menu bar "Exit to DOS" might display for an executable, but "Exit to FoxPro" would also be displayed if the application was being run from the command window. The "Exit to FoxPro" menu bar comment snippet would contain the text:

```
*:IF ! "EXE"$vers()
```

4c. I also display certain menu bars depending on the permission level of the user. Each user has a permission level, and that value is stored to a system-wide memory variable when the user logs on.

Each menu bar is restricted to a certain permission level - for example, access to the User Maintenance menu bar is restricted to users with a permission level of "1". Thus, the comment snippet of this menu option would contain the text:

```
*:if m.gcPermLevel = "1"
```

and the menu bar would only be displayed if the user's permission level is "1".

Other stuff

The documentation for GENMENUX runs 21 pages (which can be squished down to 12 if you don't believe in margins and use a font detrimental to your eyesight). There are 12 CONFIG.FP level directives, 33 menu setup level directives, and 25 menu bar comment snippet level directives.

Obviously, we've only touched on a few of them. Once you're comfortable with the process, there are several other major areas to explore.

First, colors. You can control the default color of pads and popups through the *:PADCOLOR and *:POPCOLOR directives. You can also define color schemes and assign different color pairs to individual pads or bars with the *:COLOR and *:COLORSET directives.

Next, you've got a wide range of directives to handle popups. For instance, *:POPFIELS and *:POPFIELDS create menus that offer popups of specific fields (in the current table) or files (matching a skeleton). *:POPCOMMAND identifies the action to be performed after these two. *:POPTITLE creates a popup with a title - and this title can be a constant or a memory variable.

A third area is libraries. Suppose you have a standard Utilities menu pad that routinely includes five or six bars like Backup, Reindex, Archive, and so on. You can define a Utility library that contains all the information about the bars, and then in your new menu, just reference the library. The Utilities pad and the associated bars will be added to your menu automatically.

Four tips and tricks

1. Be sure to remove any spaces between the asterisk, the colon, and the keyword, like so:

```
*:IF <expr>
```

This syntax will bomb:

```
*: IF <expr>
```

2. Since GENMENUX adds additional overhead to the menu generation process, you will likely notice a performance hit when generating menus with several directives.

If your menus are large (that's a subjective call), some of the operations that require intensive processing of each record in the menu table will take a long time. In particular, you may think that FoxPro/GENMENUX has locked up when the "Processing... 30%" thermometer bar is displayed.

Instead of reaching for the Ctrl-Alt-Delete combination, open up Debug and put the following expressions in the left side of the window:

```
SYS(16)
RECN()
```

As GENMENUX runs, the evaluation of these expressions will be updated. You'll see the name of the current procedure across from SYS(16) and the number of the record being processed across from RECN().

3. When you're using the *:IF directive, watch multiple conditions. Because the condition is preceeded with the NOT, you may need to group them with parens:

```
*:if (m.gcPermLevel = "1" or m.gcNameUser = "SUPERVISOR")
```

will produce Cleanup code like so:

```
IF NOT (m.gcPermLevel = "1" or m.gcNameUser = "SUPERVISOR")
   RELEASE BAR 7 OF UTILITIES
ENDIF
```

4. Again, when you're using the *:IF directive, you may find that your menu ends up with two separator bars next to each other because all the bars between them have been released. You can also use the *:IF directive in the comment snippet for the appropriate separator bar to avoid this less-than-elegant display.

Where to find it

GENMENUX has been placed in the public domain. It can be found on this month's Companion Disk or downloaded from CompuServe's FoxForum.