Issue Date: FoxTalk September 1998

The Visual FoxPro Screen Image Printer

Art Bergquist
abergquist@visionpace.com

Wouldn't it be great to have a screen image storage and printing program that you could ship -- royalty-free -- with your applications? Your customers could use this tool as a centralized mechanism to report defects as well as document and illustrate desired enhancement requests. Art has wanted this for a long time, and with the help of a co-worker's idea, he put this program together.

The underlying mechanism of capturing a screen image is straightforward -- something that you've probably done a thousand times. Or perhaps not. Before you can print a screen, you must first take a snapshot of the screen. Windows Help says the following regarding copying the window or screen contents:

- To copy an image of the entire screen, press <PrintScreen>.
- To copy an image of the window that's currently active, press <Alt + PrintScreen>.

I've found that while most people know about PrintScreen (jeepers, there's a key on the keyboard with this label), a surprising number aren't aware of the Alt-PrintScreen combination. But now you know. Doing either of these things places the requested image onto the Windows clipboard.

Windows Help then goes on to explain what to do with the image that's been placed onto the Windows clipboard: "To paste the image into a document, click the Edit menu in the document window, and then click Paste."

So far, so good. Combining the above with a thought from my co-worker Mike Childers and a little bit of sleight of hand with the Windows API, I had all the raw information to make a FoxPro Screen Image printer. Mike's suggestion was to put a screen image into a general field of a table and then print that field through a report. The following program is the result.

```
*-----------------------------------------------------
* Program Name: PRINTSCR.PRG (based on an idea by
*               Mike Childers)
*       Author: Art Bergquist
*         Date: April 6, 1998
*      Purpose: Print Screen utility
*
*  DO PrintScr WITH .T.
*  or
*  1. Press <PrintScreen> or <Alt+PrintScreen> for
*     the screen you want to capture
*  2. DO PrintScr
*-----------------------------------------------------

LPARAMETERS tlAutoScreenPrint
LOCAL lnSelect

IF tlAutoScreenPrint
  * "Press" <PrintScreen> or <Alt+PrintScreen>
  * for the user
  DECLARE INTEGER keybd_event IN Win32API ;
       INTEGER, INTEGER, INTEGER, INTEGER
  VK_SNAPSHOT = 44

  * Copy the active application window to the clipboard
  * (equivalent to <Alt+PrintScreen>)
  =keybd_event(VK_SNAPSHOT, 0, 0, 0)
ENDIF

lnSelect = SELECT()

DEFINE WINDOW PrintScr FROM 500, 500 TO 600, 600
```

```
ACTIVATE WINDOW PrintScr

* Ensure that a PrintScreen cursor doesn't already exist
USE IN IIF(USED('PrintScreen'), 'PrintScreen', 0)

CREATE CURSOR PrintScreen (ScreenPrint G, Notes M)
SELECT PrintScreen
APPEND BLANK

* <Ctrl-V} = Paste the contents of the Windows clipboard
*            into the ScreenPrint general field
* <Ctrl-W> = Save changes to and close the ScreenPrint
*            general field
KEYBOARD '{Ctrl+V}{Ctrl+W}' PLAIN CLEAR
MODIFY GENERAL ScreenPrint WINDOW PrintScr
IF EMPTY(ScreenPrint)
  && i.e., No OLE object in the General field
  =MessageBox('Nothing in the Windows clipboard to ;
     print!', 0 + 48, _Screen.Caption)
ELSE
  * PrintScr was designed to be run from either a VFP
  * application or the Command Window
  * Note: PrintScr form is Modal
  DO FORM PrintScr
ENDIF

USE IN PrintScreen

RELEASE WINDOW PrintScr

* Restore to the original work area
SELECT (lnSelect)
```

Under the hood

Let's analyze the program. I first check whether .T. was passed as a parameter; if so, the user (or developer) wants the program to automatically press <Alt+PrintScreen> for him or her. This is done via an API call: If .F. was passed as a parameter or PrintScr was called niladically (that is, with no parameter), then the user must first press <PrintScreen> or <Alt+PrintScreen> before running the program.

Next, I defined a window that the user will never see in order to capture the screen behind the scenes. Then I created a cursor that contains a General field and a Notes field and automatically adds a blank record so that I have a record in whose General field I can store the image.

The key to the program (no pun intended) is the combination of the KEYBOARD and MODIFY GENERAL commands. The MODIFY GENERAL command "opens editing windows for general fields in the current record. When an editing window is open, you can insert, modify, or delete an OLE object." (I originally looked at the APPEND GENERAL command but decided against it because the OLE object to be imported must already exist in a file.)

Putting all of this together, I came up with the following:

```
KEYBOARD '{Ctrl+V}{Ctrl+W}' PLAIN CLEAR
MODIFY GENERAL ScreenPrint
```

<Ctrl+V> pastes the contents of the Windows clipboard into the ScreenPrint general field. <Ctrl+W> saves the changes to and closes the ScreenPrint general field. So, KEYBOARDing <Ctrl+V> and <Ctrl+W> in conjunction with the MODIFY GENERAL command automates the process of opening the General field, pasting the screen image (that either you'd previously taken, by pressing <PrintScreen> or <Alt+PrintScreen>, or that the program had automatically taken) and, finally, closing the General field.

Then I added the clause "WINDOW PRINTSCR" to the MODIFY GENERAL command. This causes the General field editing window to take on the characteristics of the PrintScr window; as a result, the user won't see the opening of, pasting into, and closing of the General field.

At this point, I check to see whether the General field is empty (if it is, it tells me that no OLE object is in the general field and that I can, as a result, let the user know that there's nothing in the Windows clipboard to print). This step really only applies when calling the PrintScr program in "non-Automatic-PrintScreen" mode (that is, when the developer doesn't call PrintScr with a .T. parameter).

If the General field has been filled in, I could go ahead and run a report form against the PrintScreen cursor. Instead of immediately doing that, though, I decided to provide additional flexibility by calling the PrintScr form, as shown in **Figure 1**.

Figure 1. The PrintScr form allows the user to view the captured screen and to add notes about it.

This form contains an OLEBoundControl (in the upper half of the form) whose ControlSource property is set to "PrintScreen.ScreenPrint." In the lower half of the form, there's an edit box whose ControlSource property is set to "PrintScreen.Notes." This Notes field allows your user to enter notes relevant to the screen image in the upper half of the form. These notes can be an enhancement request, a bug report, an opportunity for your clients to wax poetic, a compliment, whatever!

After entering any optional notes, the user can either select <Print> or <Exit>. <Exit> confirms whether you want to exit without printing the screen image (so you don't accidentally lose all those critical notes you arduously entered). If you select <Print>, PrintScr displays the report in the Report Preview window.

The end user can either click on the printer icon, or he or she can exit at this point by simply closing out the preview window. (Of course, you can send the report straight to the printer by issuing "REPORT FORM PrintScr NOEJECT NOCONSOLE TO PRINTER" in lieu of "REPORT FORM PrintScr PREVIEW.")

Running PrintScr in an application

To set up a hot key to run PrintScr from within the application, issue the following:

```
ON KEY LABEL SHIFT+f7 DO PrintScr.EXE WITH .T.
```

Then PrintScr automatically captures the active window. If you want the program to print the screen you specify (you must first press <PrintScreen> or <Alt+PrintScreen>), you'd change the preceding code to read:

```
ON KEY LABEL Shift+F7 DO PrintScr.EXE
```

There you go! A VFP screen image printing program that you can embed as part of your error handler.

Enhancing PrintScr

No sooner was this program done than I started coming up with some more ideas. Depending on your needs, you might want to include some or all of these:

- Enable PrintScr to be called standalone (that is, outside of either a running VFP application or the Command Window).
- Enable the user to specify whether to preview and/or print from the PrintScr form itself.
- Prevent the program from being called recursively (that is, calling PrintScr while running PrintScr).

I was thinking it would be nice if we could get a manufacturer to put a new key on the keyboard called FoxSPrint, but I won't hold my breath. I hope you enjoy this program, and I invite your feedback.