Memos Under Glass

CURT SCHROEDERS

Protecting the contents of a memo while allowing it to be viewed by the masses

Working with memos in dBASE has always been an interesting challenge. With the most recent release of dBASE IV though, the memos are much easier to work with. There are functions and commands that do everything from importing a text file to plucking out a single line of the memo for you to view. But there's one thing I'd still like to see: the ability to display and scroll a memo while suppressing the ability to edit.

There is no way internal to dBASE IV to create a screen form that displays memo data in a window that cannot be edited. The workaround for this that is often used is to set the **Edit options: Accept Value When** setting for the memo field to FALSE. When you do this, the memo can still be entered and modified, but when you try to save the changes, the error message "Can't write to read-only file" pops up. So this method works but the result can be very confusing to the person getting the message.

The task then is to create "look but don't touch" memos in your screen forms — and it's surprisingly simple to accomplish. To demonstrate this concept, we will be creating a small database called Customer.DBF. It will consist of the following structure:

```
Field      Type        Length     Index

Custname   Character     20         N
Notes      Memo          10         N
```

After creating this database structure, enter some data into the first two records. Remember that when you are typing text into a memo field, let the memo field do the wrapping for you. This way you can set the width of the memo to any valid value and the data won't chop off at various places in the text.

Now we'll create a simple screen form that displays the first four lines of the memo field on the screen. We'll do this with the help of calculated fields which are not updatable. But a few explanations first.

Memo Field Functions

We will be using the MLINE() function which is explained in the Functions section of *Language Reference*. The MLINE() function returns the line of the requested memo that is specified in the function. Another function, MEMLINES(), returns the number of lines in the specified memo field. This function will be used in a specialized program (see page 31). One very important thing to note about these two functions is that they are dynamic and will return different results based on the value that SET MEMOWIDTH is set to. For example, consider a memo field called Quotes which contains the following text:

```
Now is the time for all good men to come to the
aid of their country.
```

If the memowidth is set to 14, the following output will result:

```
Now is the time
for all good
men to come to
the aid of
their country.
```

Knowing this, then the following output will also be true:

```
.? MLINE(Quotes, 2)
for all good

.? MEMLINES(Quotes)
        5
```

So both of these functions will return a value based on the width you specify, which essentially means you have more control over the display of your memo field.

Creating the Screen Form

Now, let's create a simple screen form that will use these concepts. Make sure the Customer database is in use and then proceed to create a new screen form. Once in the screen design work surface, follow these steps to create the screen form.

1. Position the cursor on row 2, column 3 (as designated by the status bar) and press F5 (Add Field). From the selection list, highlight Custname and press Return, then Ctrl-End to save the field with its default settings.

2. Create a box by choosing the **Box: Double line** from the Layout menu and anchor the upper left corner of the box on row 9, column 15 by pressing Return. Make the bottom right corner of the box row 14, column 64; pressing Return again to establish the location of the box on the screen.

3. Move the cursor to row 10, column 16 and press the F5 (Add field) key. From the field list, press the right-arrow key to move over to the Calculated column and highlight <Create>. Highlight the Expression prompt. Pressing Return will allow you to enter the expression for this field:

   ```
   MLINE(Notes, 1)
   ```

   After entering the expression, save the field by pressing Ctrl-End. The cursor should be positioned right after the block of X's representing the calculated field.

   Note that we need not designate a name for this calculated field. In dBASE IV, if no name is specified for a calculated field, then a unique one will be assigned at the time of code generation. In the long run, this is a saving grace and will allow us to skip modifying several calculated field names.

4. Press the left-arrow key and highlight the calculated field: To size the field so that it spans the width of the memo box, press Shift-F7 and, using the right-arrow key, stretch it to the right side of the box (row 10, column 63), then press Return.

5. To copy the calculated field down to the next three lines, highlight the calculated field and press F6 (Extend Select). Now that the field is selected, press Return then F8 (Copy) to copy the field to the three lines below.

6. Change the copied fields to reflect the correct memo lines by highlighting each of them and pressing F5 to change the expression. For example, the expression in the original calculated field was MLINE(Notes, 1). The expression of the calculated field on the second line should be MLINE(Notes, 2) and so on.

You now have a form in which you can see the first four lines of the memo field while entering data in the screen. Save the form with the name of Customer by pressing Ctrl-End to save it. After the form has been saved, highlight the form in the Control Center and press F2 to see it.

Notice that when you view this form there is no way to view more than the first four lines. Therein, lies the limitation of such a method and justifies the need for some special coding. Wouldn't it be nice to have the option to highlight that box and scroll up and down through the memo?

Expanding the Concept: ViewMemo.PRG

The code on the opposite page: ViewMemo.PRG uses the concept of a popup menu that contains all of the lines of the memo field, making a scrollable memo field. It does not *appear* as if it were a popup but that's really the whole point: taking the concept and using it in a totally different way. Here are a few pointers when using ViewMemo:

- Since ViewMemo performs an EDIT internally, you could use it in place of an EDIT command. If you have a menu option that edits your records, using the ON KEY LABEL would not be appropriate and you would instead perform a DO ViewMemo.
- ViewMemo reassigns the ON KEY status of F10 so, if this conflicts with a system you already have set up, assign the ON KEY LABEL statement to another key.
- The program has been designed to use macro substitution. On some slower processor machines, this may result in speed degradation. You can modify the program and substitute the literal name of a memo in place of the macro &fldname.
- One last little bonus in version 1.5: You can click the mouse on the top or bottom border of the memo and traverse the memo line by line. In either case, when you reach the top or bottom of the memo, the memo wraps to the opposite end. ■

ViewMemo.PRG

```
* Author - Curt Schroeders
* Displays the contents of a memo field using a popup menu.
* A database and format file must already be in use.
PARAMETERS fldname
IF TYPE("&fldname") <> "M"
  RETURN
ENDIF
SET TALK OFF
ON KEY LABEL F10 DO P_Viewer
mwidth = SET("MEMOWIDTH")
SET MEMOWIDTH TO 40
SET MESSAGE TO "Press F10 to view contents of memo"
EDIT

RETURN


PROCEDURE P_Viewer
  IF MEMLINES(&fldname) = 0
    RETURN
  ENDIF

  SAVE SCREEN TO S_Save

  DEFINE POPUP MemoTest FROM 9,15 TO 14,64;
       MESSAGE "Press Escape to return to editing..."

  x = 1

  DO WHILE x < MEMLINES(&fldname) + 1
    DEFINE BAR x OF MemoTest PROMPT MLINE(&fldname, x)
    x = x + 1
  ENDDO

  ON SELECTION POPUP MemoTest DEACTIVATE POPUP
  ACTIVATE POPUP MemoTest
  RESTORE SCREEN FROM S_Save
  SET MEMOWIDTH &mwidth
  SET MESSAGE TO
RETURN
```