April, 2005

Visual FoxPro 9/8/7

Easier Date Entry

Give users a control that makes entering dates as easy as it is in Intuit Quicken.

By Tamar E. Granor, technical editor

As I've written previously, I think the user interface for Intuit's Quicken is an excellent piece of work. It's full of little things that make it much easier to enter what I find to be incredibly boring data: my personal accounting information. Over the years, I've tried to bring some of those features into my own applications. (For example, see my article "Add QuickFill to Your Combos" in the September 1998 issue.)

One area where Quicken shines is date entry. Quicken's date text box lets you modify the displayed date in a variety of ways and offers a calendar for selecting a date. The date text box lets you use the + and - keys to change the entry one day at a time, and it accepts a number of shortcuts, such as "T" for today's date, "M" for the first of the month, and so on. Of course, you can simply type the desired date, as well.

It's possible to build a similar control with native Visual FoxPro code (see the sidebar), but the ActiveX Date and Time Picker control that comes with VFP already includes a drop-down calendar. It doesn't take much code to make it respond to the appropriate keystrokes.

To provide Quicken-like date manipulation, I built a custom class with all the necessary code. This article shows you what you need and discusses a few issues that have to be resolved. Figure 1 shows the custom control with the calendar expanded.

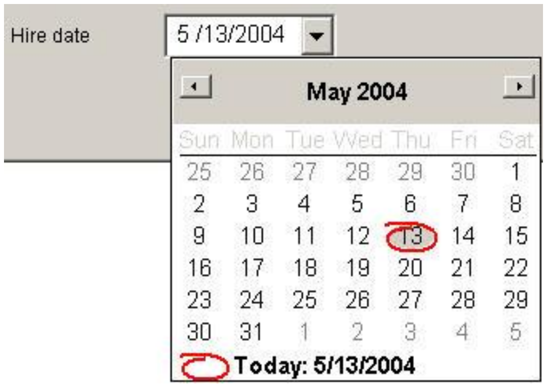

Figure 1: Date and time picker -- This control makes it easy to enter or choose a date.

Set up a Date and Time Picker subclass

My oleQuickenDatePicker class, included on this issue's Professional Resource CD, is based on (but not subclassed from) the acxDTPicker class, which Marcia Akins published in *MegaFox: 1002 Things You Wanted to Know about Extending Visual FoxPro*. (I included Marcia's code in this article with her permission.) In particular, the Date and Time Picker control has problems when you bind it to a field with an empty value; Marcia's code works around that limitation. The class has custom cControlSource and tDefaultValue properties. Code in Init copies the ControlSource to the cControlSource property and unbinds the control.

Code in Change and Refresh calls the custom UpdateControlSource and SetValue methods, respectively. The Change method calls UpdateControlSource whenever the user changes the value of the control; UpdateControlSource stores the new value to the actual control source (whose name is stored in cControlSource), like this:

```
LOCAL lcAlias
WITH This
   lcAlias = UPPER(JUSTSTEM(.cControlSource))
   IF INLIST(lcAlias, "THIS", "THISFORM")
      STORE .Object.Value TO (.cControlSource)
   ELSE
      REPLACE ( JUSTEXT( .cControlSource ) ) ;
         WITH ( .Object.Value ) ;
         IN ( JUSTSTEM( .cControlSource ) )
   ENDIF
ENDWITH
```

SetValue, called by Refresh, handles the other side of the problem, retrieving the value from the actual control source whenever the control is refreshed. The code ensures the control never contains an empty date, replacing that value with a specified default value (the tDefaultValue property):

```
LOCAL ltValue
*** Make sure the field doesn't contain an empty date
ltValue = EVALUATE( This.cControlSource )
IF NOT EMPTY( NVL( ltValue, '' ) )
   This.Object.Value = ltValue
ELSE
   This.Object.Value = This.tDefaultValue
ENDIF
```

In addition, because the Quicken-like control is only for dates, not datetimes, I used the control's ActiveX property sheet to set the date format to 1-dtpShortDate and sized the control to show just the date.

Add Quicken keystrokes

With the problem of empty dates out of the way, let's look at the code you need for Quicken-type date manipulation. You need two more custom properties: lHandled and lOldAutoYield. I'll explain how to use them later in the article.

Most of the code for emulating Quicken's behavior is in a custom method called HandleKey, which is called from the control's KeyPress method. HandleKey checks for the various keystrokes that can modify the date (shown in table 1) and processes them. All the keys that change to the first or last day of a particular period are sensitive to the current date value; if it's already an item of the specified type (such as the first day of the month or the last day of the year), pressing the key goes back or forward one more of the specified period. For example, if the date in the control is April 1, 2004, and the user presses M, the date changes to March 1, 2004. If the user presses M again, the date changes to February 1, 2004.

Table 1: Quick date manipulation -- The Quicken Date Picker handles these keystrokes.

Key	**Meaning**
+	Increase date by one day.
-	Decrease date by one day.
T	Change to today's date

Key	**Meaning**
W	Change to first day of week (based on SET FDOW value). If the date is already the first day of the week, go back one week.
K	Change to last day of week (based on SET FDOW value). If the date is already the last day of the week, go forward one week.
M	Change to first day of month. If the date is already the first day of the month, go back one month.
H	Change to last day of month. If the date is already the last day of the month, go forward one month.
Y	Change to first day of year. If the date is already the first day of the year, go back one year.
R	Change to last day of year. If the date is already the last day of the year, go forward one year.

Like many VFP controls, the Date and Time Picker has a KeyPress event, which fires when the user types any key. The code for the KeyPress method is quite simple. It calls the HandleKey method and then checks to see whether the current keystroke was handled. If so, the method sets the KeyAscii parameter to 0 to prevent the control's default behavior. (For most of the keys handled by this code, the default behavior is simply to beep.) Setting KeyAscii to 0 here is analogous to issuing the NODEFAULT command in VFP's native KeyPress method. But NODEFAULT doesn't prevent default behavior in ActiveX controls; each control has its own approach to that. The code in KeyPress is short and sweet:

```
LPARAMETERS keyascii

LOCAL lHandled

This.HandleKey(KeyAscii)
IF This.lHandled
   keyascii=0
   * Need to call Change explicitly because it's
   * suppressed by change to keyascii
   This.Change()
ENDIF

* Prepare for next keystroke
```

```
This.lHandled = .F.
RETURN keyascii
```

Here's the code for the HandleKey method, the heart of this control:

```
LPARAMETERS nKey

LOCAL dOriginalValue

dOriginalValue = TTOD(This.Object.Value)

DO CASE
CASE UPPER(CHR(nKey)) = "T"
   * Today's date
   This.Object.Value = DATE()
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "W"
   * First of week. If already on first of week,
   * go back a week.
   IF DOW(dOriginalValue) = SET("FDOW")
      This.Object.Value = dOriginalValue - 7
   ELSE
      This.Object.Value = dOriginalValue - ;
         MOD(DOW(dOriginalValue) - SET("FDOW"), 7)
   ENDIF
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "K"
   * End of week. If already on end of week,
   * go forward a week.
   nLastDayOfWeek = EVL(SET("FDOW")-1,7)
   IF DOW(dOriginalValue) = nLastDayOfWeek
      This.Object.Value = dOriginalValue + 7
   ELSE
      This.Object.Value = dOriginalValue + ;
         MOD(nLastDayOfWeek-DOW(dOriginalValue), 7)
   ENDIF
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "M"
   * First of month. If already on first of month,
   * go back a month
   IF DAY(dOriginalValue) = 1
      This.Object.Value = GOMONTH(dOriginalValue, -1)
   ELSE
      This.Object.Value = dOriginalValue - ;
         DAY(dOriginalValue) + 1
   ENDIF
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "H"
   * Last of month. If already on last of month,
   * go forward another month.
   IF DAY(dOriginalValue + 1) = 1 && last of month
```

```
      This.Object.Value = GOMONTH(dOriginalValue + 1, 1) ;
         - 1
   ELSE
      This.Object.Value = GOMONTH(dOriginalValue - ;
         DAY(dOriginalValue) , 1)
   ENDIF
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "Y"
   * First of year. If already on first of year,
   * go back a year
   IF MONTH(dOriginalValue) = 1 AND DAY(dOriginalValue) = 1
      This.Object.Value = GOMONTH(dOriginalValue, -12)
   ELSE
      This.Object.Value = DATE(YEAR(dOriginalValue), 1, 1)
   ENDIF
   This.lHandled = .T.

CASE UPPER(CHR(nKey)) = "R"
   * Last of year. If already on last of year,
   * go forward a year.
   IF MONTH(dOriginalValue) = 12 AND ;
      DAY(dOriginalValue) = 31
      This.Object.Value = GOMONTH(dOriginalValue, 12)
   ELSE
      This.Object.Value = DATE(YEAR(dOriginalValue), 12, 31)
   ENDIF
   This.lHandled = .T.

CASE nKey = 43 AND NOT This.lHandled && "+"
   This.Object.Value = dOriginalValue + 1
   This.lHandled = .T.

CASE nKey = 45 AND NOT This.lHandled && "-"
   This.Object.Value = dOriginalValue - 1
   This.lHandled = .T.

OTHERWISE
   * Do nothing
ENDCASE
```

Suppress normal behavior

When the calendar isn't open, one portion of the date (the month, the day, or the year) is highlighted. By default, pressing + or – increases or decreases that portion by 1. As table 1 indicates, the entire date should change by one day rather than changing a single piece of the date. If the user presses + or – on the alphabetic portion of the keyboard, the KeyPress event fires and calls HandleKey. However, the control handles the + or – on the numeric keypad earlier, in its KeyDown event. So, you need code in KeyDown to prevent the default behavior:

```
LPARAMETERS keycode, shift

LOCAL dCurrent

* Have to change built-in behaviors here. KeyPress is too late.
* This code suppresses the normal behavior of + and - on the
* numeric keypad. Then, KeyPress can handle them along with the
* + and - from the alphabetic keyboard.
DO CASE
CASE INLIST(KeyCode, 107, 109) && + or -
   keycode = 0
ENDCASE

RETURN keycode
```

To make all this code work, you have to set the AutoYield property of _VFP to False. The custom lOldAutoYield property saves the original value of AutoYield. Then, code in Init sets it to False, and code in Destroy restores the original value.

Limits

There are a couple of behaviors of the Quicken date control that I haven't been able to duplicate with the Date and Time Picker. In Quicken, the keystrokes in table 1 work even in the drop-down calendar. I haven't found an event that fires when the user presses a key while the calendar is open. So, VFP ignores all the special keystrokes when the calendar section is open.

In addition, I've found no way to highlight the entire date rather than just one portion (day, month or year).

Put the Quicken date control to work

Using the control is as easy as dropping it on a form. You might want to set its ControlSource or use the ActiveX property sheet to specify an initial value.

This issue's Professional Resource CD contains QuickenDateDemo.SCX, a form shown in figure 2 that includes two Quicken date controls for you to test.

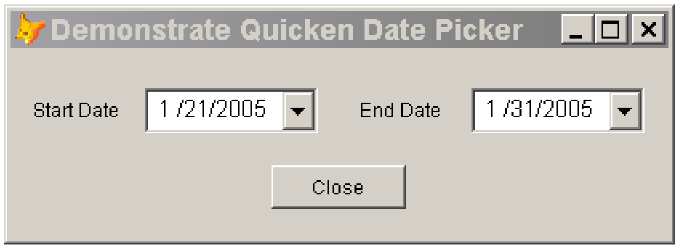

Figure 2. Using the Quicken date control -- After you create the class, all you have to do is drop the control onto a form, and all the Quicken date functionality is there.

Whether they've worked with Quicken or not, users will love having its many ways to set the date available in your VFP applications.

Sidebar: Built-In Date Manipulation

Although they don't support the whole set of keystrokes Quicken does, VFP text boxes let you use + and - to change by one day at a time. This functionality is available when the ControlSource for the text box has type date or datetime. However, the the entire date must be selected before the keystroke, so it's a good idea to set the Format property to K.