Who Needs Custom Menus?

WILLIAM HALPERN

Tired of assembling and debugging custom menu systems that ultimately prove unreliable? Here's a solution that simplifies access to application tasks by placing them on a single screen.

STARTING with Q&A 4.0 for DOS, you can create and use custom menus in addition to adding up to six new selections on the Q&A Main menu. You can even replace Q&A's menu system with a custom system. You can add to, re-create, or replace any Q&A menu with one that performs regular Q&A functions or runs custom procedures by executing macros.

Creating a stable custom menu system, however, can be a daunting task. Replacing Q&A's entire menu system is a complex and tedious process, likely to wind up blocking access to otherwise useful Q&A features. Custom menus designed to work with regular Q&A menus are further complicated because your custom menus can be used only at the proper points in Q&A. Run their selections from the wrong location, and you put your data at risk. On top of all this, Q&A doesn't like mixing custom menu driven operations with regular keyboard macros—it sometimes "forgets" where it is and executes the macros improperly.

When implemented with care and properly used in the system, custom menus are the greatest thing since sliced bread for the Q&A developer. I've used them in countless applications with great success. But I've also had to rescue clients from custom menu systems that worked unpredictably.

5.0 to the rescue

Then along came Q&A 5.0. Using the new @Msgbox, @Askuser, @Exit,

Continues on page 3

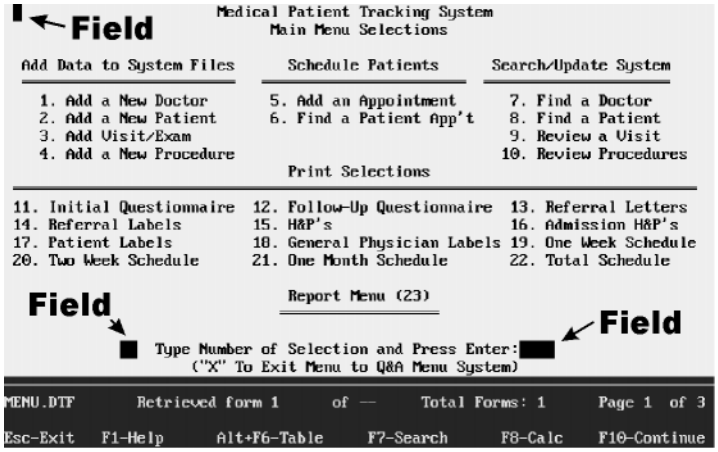

Figure 1. Page 1 of a "custom menu system" that gives users access to the entire application from a single convenient screen. The three dark rectangles are fields.

The Quick Answer™
The Independent Guide to Q&A® Expertise

June 1996
Volume 7, Number 6

Editorial

Tom Marcellus, editor

You'll love Colorado Springs, Colorado, in October, especially if you're there to attend the National Q&A User Group's annual weekend bash and *The Quick Answer*'s day-long Q&A Master's Seminar. Details in this issue.

Since Q&A 4.0 for DOS arrived on the scene, so much has been written on building, coddling, and straightening out custom menus and custom menu systems that they've become a kind of bugaboo. On the one hand, they let you organize, display, and run application-specific macros, so they can be enormous productivity boosters. On the other, you have to design them just so, and you hear reports of by-the-book custom menus going haywire when you least expect it.

As **Bill Halpern** explains in this month's lead article, Q&A 5.0 gives you an out. Depending on the application, you can display and run dozens of macros from a single application control screen. His inventive approach is worth a look, even if you're still having a love affair with custom menu systems.

Among Q&A 5.0's new features you'll find powerful commands to display pick-lists of items from external lookup databases. You can get yourself into a pickle, though, when the lookup database contains scads of records. You've waited way too long for the list to appear—"is it going to appear?"—but you can't cancel the command. I'll show you how to restrict the range of records that appears on your pick-lists in a way that simplifies, not exacerbates data entry.

Last month, Bill Halpern teamed up a modem, a couple of batch files, and Q&A's @Shell command to make Q&A auto-dial the phone number in a database record. **Gordon Meigs** takes it a step further with a handy point-and-click technique that tells Q&A which of *several* phone numbers in the record to dial.

Some multi-step processing tasks involve so many variables that you shudder at the thought of entrusting the entire procedure to one macro. **Jeff Nitka** shows you a clever "divide and conquer" technique that loops a macro the exact number of times necessary to process each set of variable retrieval parameters. **QA**

Due to a goof at the bindery, you might have received the May issue with several pages out of order. If so, open the newsletter to expose the staples and pry up the points with a letter opener or knife. Remove the inside sheets that are out of order, reinsert them in the proper order, then press the staple points back down. We apologize for the inconvenience, but heck, you now know how to bind a newsletter!

The Quick Answer™

*The Independent Guide to **Q&A**® Expertise*

Editor Tom Marcellus
Publisher Michael Bell
Copy Editor Laurie Moloney
Production Editor Paul Gould

The Quick Answer (ISSN 1052-3820) is published monthly (12 times per year) by Marble Publications, Inc., 9717 Delamere Ct., Rockville, MD 20850.

Cost of domestic subscriptions: 12 issues, $79; 24 issues, $142. Outside the U.S.: 12 issues, $99; 24 issues, $172. Single copy price: $10; outside the U.S., $12.50. All funds must be in U.S. currency. Back issues are available upon request, for the same price as a single copy.

Second-class postage paid at Rockville, MD. POSTMASTER: Send address changes to *The Quick Answer,* PO Box 9034, Gaithersburg, MD 20898-9034.

Address editorial correspondence, @HELP questions, or requests for special permission to: Marble Publications, Inc., *The Quick Answer,* PO Box 9034, Gaithersburg, MD 20898-9034. Phone 800-780-5474 or 301-424-1658. Fax 301-424-1658. CompuServe 73370,1575. Prodigy NEPY97A.

For Q&A technical support, call Symantec: 541-465-8600.

Who Needs Custom Menus?

Continued from page 1

@Save, @Color, and—most importantly—@Macro commands, I found a way to circumvent all my old custom menu problems.

@Msgbox lets you pass messages to the user. @Askuser lets you prompt for instructions on how to proceed. @Exit and @Save allow you to control when and how a user exits a record. The @Color command, along with the new custom color palette, lets you create invisible fields, and @Macro lets you conditionally execute a macro that can perform just about any task you want. Combining these features, you can create a "custom menu system" for an application that looks like **Figures 1** and **2**.

Those with sharp eyes will recognize that this "custom menu" looks suspiciously like a database record. That's exactly what it is—a one-record database that can be changed only by the developer. In this case, it serves as the application Main menu for a medical patient tracking system that lets users run all the tasks in the application that would otherwise have to be run from a custom menu system. The beauty of it is that it makes for smooth, seamless use of the application while keeping Q&A's standard menus intact. And by performing all the application's tasks via programmed @Macro commands, it overcomes the inherent problems associated with custom and replacement menu systems.

How it works

The trick to this record-based menu system is making it work like a real replacement menu system. You want to keep the user in the menu system—within the operations that the menu selections allow—as far as possible.

The macro that initializes the application displays the record-based Main menu. In this application, the *Call MENU.DTF* macro is executed by pressing the Alt-1 hot key combination. As you can see in the sample macro that follows, all the macro does is "press" Esc three times, then perform a File / Search/Update on MENU.DTF, "pressing" F10 at the Retrieve Spec to display the one and only record in the database:

```
<begdef><alt1><name>"Call<sp>MENU.DTF"<vidoff>
<esc><esc><esc>fs<home>MENU.DTF<enter><f10><enddef>
```

To exit the menu record, the following *Quit Menu* macro resaves the record, then exits to the Q&A Main menu:

```
<begdef><nokey><name>"Quit<sp>Menu"<vidoff>
<capsf10><esc><enddef>
```

The program in the one-character labeless field in the top left corner of the form (see Figure 1) makes it impossible to add another record to the database. Another field at the beginning of the second to last line holds the bulk of the programming. The field at the end of the second to last line stores the user's menu selection.

The first field is programmed like this:

```
< #1: If @Add Then { @Msgbox(
  "You Cannot Add","Records to this Menu","");
  @Exit };
  If @Update Then Goto #10
```

If the user opens MENU.DTF in Add Data mode, the program displays a custom error message that freezes the screen, then exits the database when the user presses Enter. If the user opens the database correctly in Search/Update mode, the cursor moves to the field where the user is prompted to type in the menu selection number.

For the system to work, there must be one record in the database. Because the program precludes adding a record, be sure you create and save the single record (just place an "X" in the first field so Q&A will let you save it) *before* you program the database.

The program for field #10 is as follows:

```
> #10: If #10 = "" Then Goto #10 Else Goto #11
```

This program returns the user to the selection field if it's left blank, and moves to the selection programming field (field #11) on entry of a valid menu selection number. The programming in the last field—field #11—is shown in **Listing 1**:

Listing 1. The program for field #11.

```
< #11: @Color(#11,13,5);
If #10="X" Then {#10="";@Macro("Quit Menu")};
If #10="1" Then {#10="";@Macro("Add New Doctor")};
```

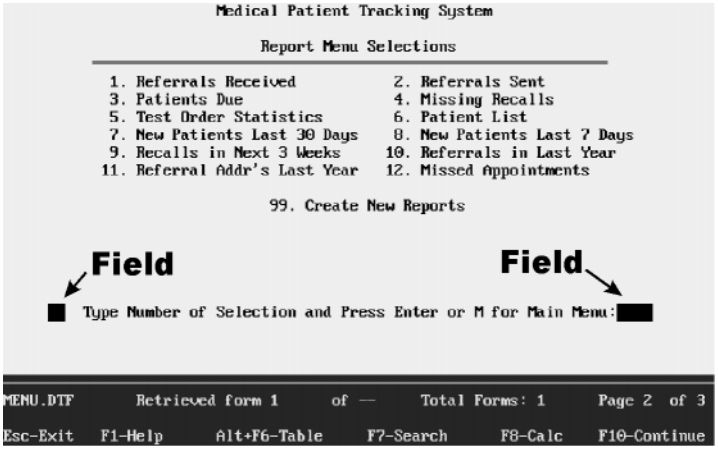

Figure 2. Page 2 of the application menu. The two dark rectangles are fields.

```
If #10="2" Then {#10="";@Macro("Add New Patient")};
If #10="3" Then {#10="";@Macro("Add New Visit")};
If #10="4" Then {#10="";@Macro("Add New Proc")};
If #10="5" Then {#10="";@Macro("Schedule Patients")};
If #10="6" Then {#10="";@Macro("Schedule")};
If #10="7" Then {#10="";@Macro("Search for Doctor")};
If #10="8" Then {#10="";@Macro("Patient Search")};
If #10="9" Then {#10="";@Macro("Latest Visit")};
If #10="10" Then {#10="";@Macro("Search Proc")};
If #10="11" Then {#10="";@Macro("Print Patquest")};
If #10="12" Then {#10="";Goto #30};
If #10="13" Then {#10="";Goto #30};
If #10="14" Then {#10="";Goto #30};
If #10="15" Then {#10="";@Macro("Print H&p Letters")};
If #10="16" Then {#10="";@Macro("Print Admit H&p")};
If #10="17" Then {#10="";@Macro("Labels")};
If #10="18" Then {#10="";@Macro("Doc Labels")};
If #10="19" Then {#10="";@Macro("One Wk Schedule")};
If #10="20" Then {#10="";@Macro("Two Wk Schedule")};
If #10="21" Then {#10="";@Macro("One Month Schedule")};
If #10="22" Then {#10="";@Macro("Total Schedule")};
If #10="23" Then {#10=";Goto #20}
```

Of course, you'll substitute your own task descriptions and macros, but your program's structure will follow similar lines.

In the menu form for the actual application, there's another page of menu subchoices for selections that require multiple or additional choices. (You can use up to 10 pages.) The *Goto #20* and *Goto #30* commands for selections 12, 13, 14, and 23 (see Listing 1) move the user to a similar selection field on the second or third page.

When the cursor enters the selection field—field #11—the @Color command (see Listing 1) changes its color to keep it invisible—only the cursor and value show. This makes the form look less like a record, and more like a custom-designed menu selection screen. Entering a valid selection blanks the selection field (to keep the record clean for its next use) and executes the associated macro that, in turn, performs the selected task.

Let's look at one of the selection macros to see how they work.

The macros

If the user wants to add a new physician to the doctor database (DOCTOR.DTF), the program clears the entry field and runs the following pre-recorded *Add New Doctor* macro:

```
<begdef><nokey><name>"Add<sp>New<sp>Doctor"<vidoff>
<capsf10>a<home>doctor<enter><enddef>
```

The macro saves and exits the MENU record to the File menu. It then "presses" *A* for Add Data (make sure you have Q&A set to Automatic Execution), *Home* to clear the filename prompt, then enters the name of the database (Doctor). A blank record appears to add the new physician.

It sounds simple enough. But there's another trick you can use to make your database menu system really sparkle. If you left it at this, when the user was finished entering one or more new record(s) in the DOCTOR database and exited, Q&A would return to the File menu, not the original MENU record. To return the user to the MENU record, you can add a programming statement like this to the last required field in each database used by the application:

```
> #999: If @Add Then {
If @Askuser("Do You Want to Add","Another Doctor?","")
   Then @Save Else @Macro("Exit Docs") }
```

Then, create the following macro:

```
<begdef><nokey><name>"Exit<sp>Docs"<vidoff>
<capsf10><alt1><enddef>
```

Here, the @Askuser command asks the user if another record is to be added. If the answer is "Yes," the @Save command saves the record and displays another blank record to be filled. If the user answers No, the program invokes the *Exit Docs* macro which, in turn, saves the record and exits, then invokes the Alt-1 macro that recalls the MENU.DTF menu record.

That does it. It might seem like a lot of work, but if you want a smooth, reliable, and seamless menu system fully tailored to your client's requirements, it's worth the time you'll invest creating it. **QA**

Bill Halpern is executive vice president of Professional Computer Technology Associates in Newtown, Pennsylvania. Bill has been designing and installing Q&A business applications for the past eight years. 215-598-8440, CompuServe 71023,356.

Quick Tip

Personal Startup Switch and Macro File

DOS

When running Q&A in a multiuser environment, you can use the -P startup switch to specify the path to the QA.CFG configuration file that stores the user's Network ID, printer setup, global options, recently used file list, and the like. What isn't well known, though, is that Q&A will also use a macro file stored in the same path as long as it's named QAMACRO.ASC. Therefore, users can have not only their own personal configuration files, but their own macro files as well. What's more, they can start Q&A with the -AL (autoload) or the -AD (autoload default) switch to specify a different macro path and file name. **QA**

Alec Mulvey, Keyword Training & Consultancy, Ascot, England.

Pop-Up Selection Lists and Large Databases

Tom Marcellus

When the lookup database contains so many records that your pick-list takes forever to appear, you need a way to control the selection range without hampering data entry.

You can simplify data entry by taking advantage of Q&A 5.0's XUserselect command to display pick-lists of items from an external database. But you'll run into a problem if the lookup database contains too many records. A mile-long pick-list can be more of a handicap than a help, and you can't wait forever while Q&A strains to compile and display it. You need to restrict the range of records on the pick-list in a way that simplifies, not complicates data entry.

With XUserselect you can't control the quantity of selections on a pick-list. If the lookup database contains 1,000 records, XUserselect will include them all. Depending on available memory, Q&A will display up to about 3,200 items on a pick-list, but unless you have a lightning fast PC, you'll wait and wait (and wait) while Q&A compiles and displays a list of that magnitude.

This is why if your lookup database contains more than a few hundred records, you're better off using the XUserselectR range command. You can then control the range of selections that appears on your pick-list, Q&A will display the list much faster, and you'll have an easier time finding the item you're after.

An example

Suppose you have a 5,000-record CUSTOMER database indexed on the Company Name field. You want to program a sales order database to display a pick-list of companies, and use your selection to retrieve the shipping and billing address, terms, and so forth. If you use XUserselect, or XUserselectR with an open range, Q&A will try to create a pick-list containing all 5,000 company names and fail. After a lengthy delay, Q&A will offer to display the first 3,200 or so—but won't display the remaining 1,800.

If, on the other hand, you use XUserselectR and specify the Company field in the current record as the starting and ending range, you can type "C" in it and press Enter to get a pick-list of all the companies that begin with the letter "C." The shorter list will display more quickly and contain only a few hundred or so company names. If you type "CR" to further restrict the range, a shorter list of companies beginning with "CR" will display even faster. Your pick-list will include Crabapple Carpets, Cracker Jack Roofing, Creaky Flooring, and so forth.

Clearly, there's an advantage to restricting the range of pick-list items from a large database, but how do you decide on—and enforce—the optimum range? The following factors should be taken into account:

- The number of records in the lookup database.
- The speed of the PC (including whether or not it's on a network, and how responsive the network is).
- The length of time you're willing to wait (or make users wait) before Q&A displays the pick-list. (For most users, a few-second delay won't seem unreasonable.)
- The number of items you want the pick-list to include.

Characteristics of names databases

Table 1 (see page 6) shows the average occurrence of the first letter of a person's last name in a database of 5,000 records, 10,517 records, and 16,665 records. My guess is the spread should be about the same for databases keyed on company names.

Take the names that begin with the letter "B." In the 5000-record database, there are 467 of them, and Q&A will display them all on a pick-list. In the database of 10,517 records, though, you've got 1,052 "B" names, and you'll wait considerably longer for the pick-list to appear. In a database of 16,665 records, you've got 1721 "B" names, and you might have enough time to do your nails.

The point is, with a large lookup database, you should force the entry of at least *two* characters to restrict the pick-list range—that is, "BA," "BR," and so forth. An extra character or two helps Q&A display the pick-list that much faster.

Now, you can't expect users to know that the quantity of "B" names requires a minimum two-letter entry, while the smaller quantity of "A" names might require just one. How do you inform users that more than one character is required, and make sure they enter them before allowing the lookup to execute?

Get the facts on your lookup database

One way is to create a lookup table with an entry for any alphabetic group that requires multiple characters (see **Figure 2**), and a program that checks it before executing the pick-list command.

Before creating the table, you'll need a report that counts the names beginning with "A," "B," "C," and so forth. Name the report something like *Count by Alpha Group* and include all the records in the lookup database. At the Column/Sort Spec, type 1,I (column 1, invisible) in the Last Name field, then create the three derived columns shown in **Figure 1**. At the Print Options screen, be sure Totals Only is set to No.

Your alphabetic group percentages should approximate those in **Table 1**. The count of names in each group will tell you what you need to know to construct your lookup table. (See Figure 2.)

Suppose your lookup database has an alphabetic spread like the 10,517-name database in Table 1. And suppose you don't want to wait more than a few seconds for the pick-list to display, and you want it to include no more than 100 or so names. In the Key column of the lookup table for the *primary* database (not the lookup database), include the letter of each alphabetic group that requires the entry of multiple characters. Then, in column 1, enter the number of letters required.

```
                              DERIVED COLUMNS

Heading: Alpha!Group
Formula No. 1  @Left(#1,1)
Column/Sort Spec: 2,AS,F(JC),R

Heading:
Formula No. 2  #2
Column/Sort Spec: 50,I

Heading: Count
Formula No. 3  @Text((#2 <> #50), @Str(@Count(#50, #2)))
Column/Sort Spec: 4,T,F(N0)

Heading:
Formula No. 4
Column/Sort Spec:

SPECIAL.DTF            Derived Columns for Counts by Alpha          of 4

Esc-Exit         F1-Help         F9-Go back to Column/Sort S      inue
```

Alpha Group	Count
A	316
B	1052
C	771
D	625
E	190
F	367
G	575
H	701
I	55
J	134
K	482
L	630
M	920
N	197
O	131
P	543
Q	24
R	512
S	1107
T	395
U	33
V	207
W	422
Y	50
Z	79
	10518

Figure 1. Derived columns for the Counts by Alpha Group report. (The inset shows the output.) If your Report Global Format Options Default to repeating values is set to No, delete the "R" code from the first derived column's Column/Sort Spec.

Average Occurrance of First Letter of Name in Names Databases

	Database1 5000 Names	Database2 10517 Names	Database3 16665 Names	Average Occurrance %	Per M*
A	156	316	530	3.1	31
B	467	1052	1721	10.1	101
C	359	771	1108	7.0	70
D	215	625	767	5.0	50
E	102	190	322	1.9	19
F	164	367	625	3.6	36
G	248	575	789	5.0	50
H	403	701	1149	7.0	70
I	38	55	104	0.6	6
J	143	134	348	1.9	19
K	238	482	859	4.9	49
L	236	630	786	5.1	51
M	447	919	1482	8.8	88
N	100	197	340	2.0	20
O	91	131	241	1.4	14
P	232	543	789	4.9	49
Q	14	24	26	0.2	2
R	251	512	882	5.1	51
S	496	1107	1792	10.5	105
T	144	395	605	3.6	36
U	14	33	90	0.4	4
V	79	207	279	1.8	18
W	268	422	845	4.8	48
X	0	0	1	0.0	0
Y	29	50	75	0.5	5
Z	45	79	109	0.7	7

*Per 1000 records

Table 1. Typical occurrence of the first letter of a name in a 5,000-record, 10,517-record, and 16,665-record database.

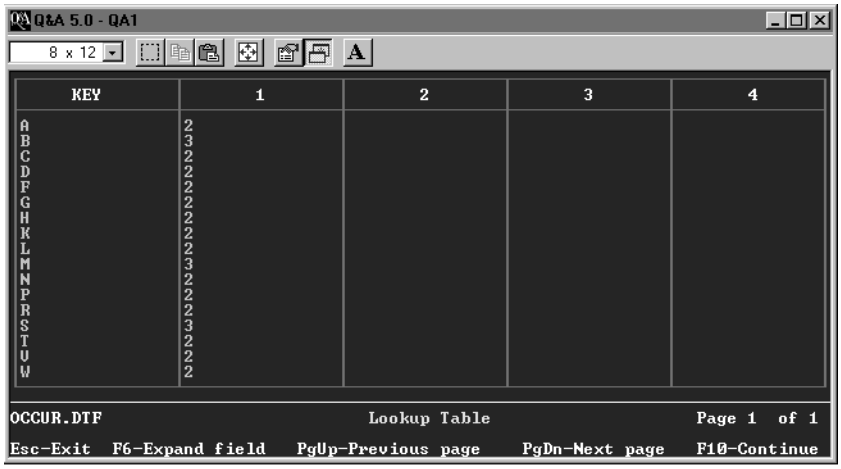

Figure 2. A sample lookup table for a 10,000-record lookup database.

Looking at the table shown in Figure 2, you'll notice that the letters E, I, J, O, Q, U, X, Y, and Z aren't there. These relatively small alphabetic groups (at least in a typical 10,000-record names database) require only the first letter of the last name to display the pick-list, so there's no need to include them in the table.

Now look at Table 1. The 10,517-record column indicates 316 "A" names, and 1,052 "B" names. So, in the lookup table for the "A" names, you specify "2," and for the "B" names you specify "3." This way, to display a pick-list of "A" names, you'll have to enter the first *two* letters of the name. For "B" names, the first *three* letters will be required. And your program, as you'll see, will enforce the rule.

Primary database design and programming

All that's left to do is program your primary database (your sales order or invoice database) to set the range and display the pick-list. **Figure 3** shows a primary database that, for illustration purposes, includes just the customer's name and address fields, and **Listing 1** shows a sample program for its Last Name field.

Listing 1. Sample program for the Last Name field in the primary database.

```
> If Last Name = ""
  Then {
  @Msg("Type first letter of Last Name and press ↵");
  Goto Last Name };

  If @Len(Last Name) < @Lookup(@Left(Last Name, 1), 1)
  Then {
  @Msg("This name range requires the first "
       + @Lookup(@Left(Last Name, 1), 1) +
       " characters of last name");
  Goto Last Name }

  Else If @Add Then
  XLookup(
  "CUSTOMER", @XUserselectR("CUSTOMER",
  "List Code", "List Code",
   Last Name, Last Name),
  "List Code", "First", First Name,
  "Last", Last Name, "Street", Street,
  "City", City, "State", State, "Zip", Zip)
```

How the program works

During data entry in the primary database, if the Last Name field is left blank, the program displays an informative message and returns to the field. If the field isn't empty, but the number of letters—*@Len(Last Name)*—is less than the number specified for the alphabetic group in the lookup table, a message tells you how many letters are required for the

Lookup Database Design and Programming

You want to ensure your pick-list includes all the names in the lookup database. Account numbers or customer IDs might be unique, but they aren't likely to tell you who the customer is.

If your lookup database contains names, and you have fields for the first name, last name, and ZIP code, you can add a field to the database to contain a unique pick-list code comprised of those three elements.

After adding the List Code field to the database, make it read-only and Speedy/Unique. Then, Mass Update the database with an Update statement like this in the new List Code field:

```
#1 = Last Name + ", " + First Name + @Str(Zip)
```

This way, the List Code field for customer *Lisa Carter* in ZIP code *37688* will contain *Carter, Lisa37688*.

To create the same List Code entries for records you later add or edit, you can use an on-record-exit program like this:

```
If @Modified Then
List Code = Last Name + ", " + First Name + @Str(Zip)
```

Because Q&A uses only the first 16 characters to determine if a value is unique, this scheme isn't foolproof (you could have two Lisa Carters in ZIP code 37688, but your pick-list will include only one of them), though you should find it workable for most applications.

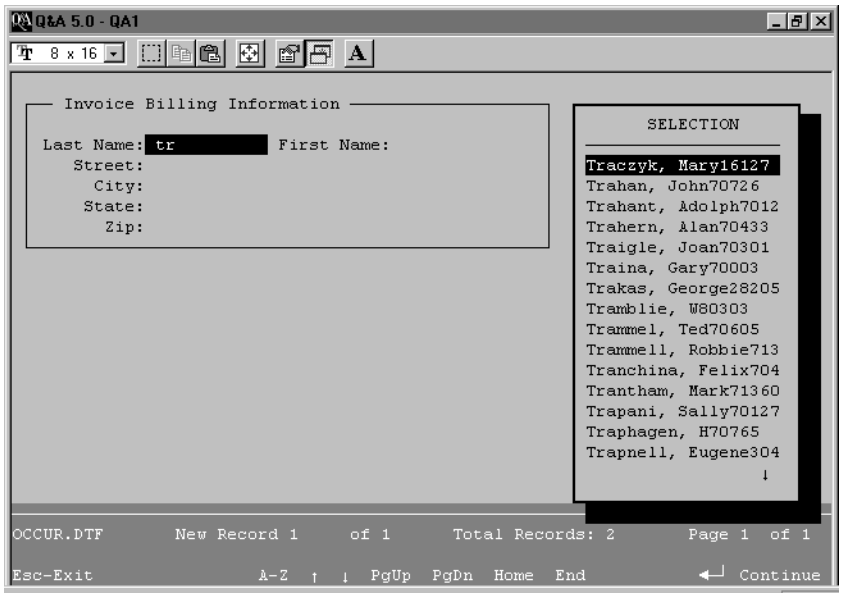

Figure 3. Selecting the customer from the pick-list. Here, the "T" names required entry of the first two characters of the last name, so the user entered "Tr" to display a pick-list of all the customers whose last names begin with "Tr."

alphabetic group you want to display, and the cursor returns to the field so you can supply them. (The program doesn't prevent you from voluntarily entering several characters right off the bat to restrict the pick-list.) See the sidebar, "Making Data Entry Even Easier."

When the program decides that the number of letters in the Last Name field is sufficient, control passes to the Else portion of the program, the pick-list is displayed, and your selection then becomes the external key value for the XLookup. (Notice that the XUserselectR command is *inside* the XLookup command.) The XLookup, then, retrieves the information for the Last Name, First Name, Street, City, State, and Zip fields. (You can lookup additional fields if you like—just add them to the XLookup command.)

If you have other alphabetically based pick-list fields in the same database, you can use columns 2, 3, and 4 of the lookup table to enter the number of characters required to restrict them, and modify their programs accordingly. **QA**

Tom Marcellus is editor of *The Quick Answer* and author of *PC World Q&A Bible*, published by IDG Books. His QuickClick Calendar Plus—a time- and activity-tracking database for Q&A 5.0—along with his QuickClick RecordFinder for Q&A 5.0, are available from Marble Publications, publisher of *The Quick Answer*.

Making Data Entry Even Easier

You'll run into a small snag with the approach in this article, but there's a workaround. Suppose you enter the first letter of the name when multiple letters are required for the name range. In this case, the program will return the cursor to the Last Name field, and it will wind up *under* the letter you've already typed. Your natural tendency will be to type the next letter, but you'll overwrite the first one if you do. Ideally, Q&A would place the cursor *to the right* of whatever letter(s) you've already typed.

You can make Q&A do this with the following macro that "presses" the End and Insert keys:

```
<begdef><nokey><name>"End/
Insert"<vidon><end><ins><enddef>
```

This way, Q&A will place the large insert cursor in the next character position—your signal that another letter is required. When you've recorded and saved the macro, simply place the @Macro command in an on-field-entry statement like the following in the Last Name field at the Navigation Spec:

```
< If @Add and @Len(Last Name) > 0
  Then @Macro("End/Insert")
```

Quick Tip

Count Your Blessings (and Your Parentheses)

Among the most common programming goofs is not having a closing parenthesis for every opening parenthesis—or vice versa. When I receive an error message after entering a section of code, the first thing I look for is a typographical error, then a missing semicolon. If I don't find any, I count my parentheses from left to right, incrementing the count by one for each opening parenthesis, and decrementing it by one for each closing parenthesis. This way, if I wind up with a positive number, I know there are too few closing parenthesis. If I wind up with a negative number, I know the opposite is true. Figure 1 shows how this technique works with a sample programming statement.

You can also use this technique to count opening and closing French braces. **QA**

Tom Marcellus

```
        1  2   1          2  3   2      1        0
If @Mid(@(#10), @Instr(@(#10), "|") + 1, 1) = "A"
Then #101 = @Left(@(#100), @Instr(#100, "|") -1)
                  1  2     1          2        1    0
```

Figure 1. Counting your opening and closing parenthesis from left to right can help you quickly spot an imbalance.

@Help

Edited by Dave Reid

Rounding Report Columns

I print reports that total my salespeoples' weekly sales amounts. Since the information isn't for accounting purposes, I'd like to round all the dollar amounts to the nearest dollar, dropping the cents. How can I do this?

Monica Wilson, Freeport, Illinois

You have several options. The simplest is to format your money columns for numbers, and set the number of decimal places to 0. You can use a Column/Sort Spec code such as the following for each money field (change the column number and other calculations as appropriate):

```
Money field: 5, ST, F(N0)
```

If you want the dollar signs, you can change the database's Global Format Options to display only whole dollars. To do this, select File / Design File / Customize a File, enter the name of your database, and choose Format Values. Press F10 to display the Global Format Options screen, set # *of Currency Decimal Digits* to 0, and press F10. You can now print your report without changes, but the new global setting will be applied to all money fields and reports in the database.

[If you don't mind the extra preparation, you can design a sales summary report that will print whole dollars with dollar signs, like this:

```
          Sold
          Last
Rep Name  Week
————————  ——-
Adler     $3284
Jones     $1746
Smith     $4740
```

Here are the Column/Sort Spec and two derived columns you'll need:

```
Rep Name: 1,AS
Sale Amt: 2,I

Heading:
Formula:     #1
Column Spec: 50,I

Heading:  Sold!Last!Week
Formula:  @Text(#1 <> #50,"$"
          + @Str(@Round(@Total(#2, #1),0)))
Column Spec: 3,F(JR)
```

The totals-only format is accomplished by suppressing the second column until there's a break in the first column, not by making it a Totals-only report at the Print Options screen.—T.M.]

Problem Opening Large Documents

After upgrading to Q&A 5.0, I encountered a serious problem. I have several large documents created in Q&A 4.0 that Q&A 5.0 won't open. I get a "Not enough memory to continue" message. What can I do? I need these documents!

Donna Dawdle, Winnetka, Illinois

The only solution I've found is to decrease the document sizes in Q&A 4.0. If you've removed or overwritten Q&A 4.0, you'll have to reinstall it to make these one-time edits, or do so on another PC running Q&A 4.0. With the documents open, you can break them into two smaller documents (using the F8 Options menu Block Operations / Move to File command) and save them to different filenames you can work with in Q&A 5.0. A quick test I performed revealed that the document limit in Q&A 4.0 is about 80 pages, dropping to about 70 pages in Q&A 5.0.

[With the document in two files, you might be able to print it acceptably using Write's JOIN command.—Ed.]

Dave Reid is a Symantec senior support analyst providing second-level assistance to the technical support representatives. He's also the co-author of *The Q&A 4.0 Wiley Command Reference*, published by John Wiley and Sons, and works as an independent Q&A consultant. PO Box 12083, Eugene, OR 97440.

Have a nagging question? Send it to ***@Help,*** *The Quick Answer, Marble Publications, Inc., PO Box 9034, Gaithersburg, MD 20898-9034* or fax to 301-424-1658. When writing, please include your name, address, and phone number, along with your Q&A version number (and whether DOS or Windows) and a detailed description of the problem. We will publish those questions we feel are of general reader interest; individual responses are not possible.

Choose Your Phone Number to Auto-Dial

Gordon Meigs

Last month, Bill Halpern teamed up a modem, two DOS batch files, and Q&A's @Shell command to make Q&A dial a phone number in a database record. This month, Gordon Meigs expands on the technique, showing how you can conveniently point-and-click to tell Q&A which of several phone numbers in the record to dial.

ONE good idea often leads to another, and Bill Halpern's auto-dialing technique in last month's *The Quick Answer* got me thinking. Though I could program Q&A to dial a database record's phone number, what would I do if the record contained several numbers, any one of which I might want to auto-dial? I found a solution that makes it a snap, and I'll show you how to do it.

Figure 1 shows a portion of a database form (at the Program Spec) containing fields for three names and three phone numbers. Field #2, the Dial field, corresponds to the Dial field Bill described in his article. Field #40, the Capture field, stores the selected phone number.

To dial any of the three phone numbers, you program the three name fields and their corresponding phone number fields so that when you click on one then click on the Dial field, Q&A copies the number to the *Capture* field and dials it.

Q&A and mouse moves

The key to this technique is the way Q&A responds to field navigation with a pointing device. If you click on Field A, then click on Field B, Q&A executes Field A's *on-field-exit* program before moving the cursor to Field A. (Compare this with using key presses to move around the form. Left arrow, Up arrow, Shift-Tab, and Home-Home-Home don't execute *on-field-exit* programs.) In this case, clicking on the name or phone number and clicking on the Dial field, copies the number to the Capture field (*on-field-exit*) then executes the Dial field's *on-field-entry* program.

Programming the database in this fashion means you can edit data in the name and phone number fields—or Tab through them—without causing the Dial field program to execute.

Programming

You can place the Dial field in any convenient location on the form. The Capture field doesn't need to be visible. You can make it a one-character labeless field, and place it in an out-of-the- way location on the form. You can also make it read-only and even set it's color scheme to make it blend with the background.

Here's the program for the Dial field:

```
< #2 = @Shell("DIAL.BAT "+ #40);
  #2 = "Talk";
  @Msgbox("Pick up phone and","press Enter to talk","");
  #2 = @Shell("RELEASE.BAT ");
  #2 = "Dial"
```

Bill's article describes the two DOS batch files you'll need to create: DIAL.BAT, and RELEASE.BAT—and explains how they work So I won't repeat that here:

```
                    Company Contacts
Name1: >#10: #40=#11          Tele1: >#11: #40=#11
Name2: >#20: #40=#21          Tele2: >#21: #40=#21
Name3: >#30: #40=#31          Tele3: >#31: #40=#31

Dial: < #2                              Capture: #40
(Click on the name or phone number,
then click here to dial it)
```

Figure 1. On-field-exit programs for the name and phone number fields. You click on any name or number field, then click on the Dial field to copy the number to the Capture field and dial it.

Gordon Meigs is vice president and general manager of Professional Computer Technology Associates of Newtown, Pennsylvania. He teaches courses and does corporate training on Q&A, and has been designing and installing advanced Q&A business applications for more than nine years. 215-598-8440, CompuServe 71023,356.

[Gordon's right—one good idea often leads to another. I wanted to be able to place a call to a business contact from any database I might be working in. So I needed a kind of "plug-and-play" solution that would display a pick-list of my contacts in any database, and auto-dial the phone number of the one I selected.

I started by adding a seven-character labeless field to a database. I made it a click-on button field named PhoneButton, formatted it T,JC for center-justified text, and assigned the value "Phone" to it via an on-record-entry program. (See my article in the February 1996 The Quick Answer.*) The external database containing the names and phone numbers was CONTACTS.DTF, so I programmed the new PhoneButton field this way:*

```
< PhoneButton =
  @XUserselectR("CONTACTS", "FullName",
                "Phone", "A", "Z");
  If PhoneButton <> "" Then {
  PhoneButton = @Shell("DIAL.BAT "+ PhoneButton);
  @Msgbox("Pick up the receiver",
          "and press Enter to talk.","");
  PhoneButton = @Shell("RELEASE.BAT ") };
  PhoneButton = "Phone"; Chome
```

This way I can simply click on the "Phone" button for a pick-list of contacts, then select the one to call. (In my case, the speedy FullName field in CONTACTS contains the last name, followed by a space, then the first name.) The XUserselectR command displays a pick-list of names, returns the phone number of the selected name to the PhoneButton field, then executes the rest of the program, dialing the number, and so forth.

It took just a few more minutes to add the same "Phone" button field to my other databases and copy the program to them. Now, no matter which database or record I'm working in, if it occurs to me to call someone, I can conveniently do so with a couple of mouse clicks, and without leaving the record I'm working on.—Ed.] **QA**

Quick Tip

Put On-Record-Exit Programming to Work

I often use on-record-exit (ORE) programming to validate a record before it's saved. Though it varies from database to database, I generally check to see if required fields have been filled properly, check the ranges of certain fields, and perform needed calculations.

Unfortunately, Q&A won't let you "Goto" a field via an on-record-exit program, so should a validation check fail, you can't return to a problem field to fix it. Working within these limits, I make the record "unusable" by using the Clear command to erase the contents of key fields. This includes all fields used by any operation, such as printing, so the "unusable" records are left out. It doesn't matter if the list of fields cleared is long. The Clear command can clear them quickly.

In addition, I notify the user when a validation fails with either a @Help screen or @Msg message. This way, the user is made aware of the fact that the record isn't valid.

Lastly, I include with the database a Retrieve Spec to remove the invalid records. Though such records are usually harmless if left saved, there's no reason to keep them. The Retrieve Spec, used in conjunction with the Remove / Selected Record operation, ensures that the database contains only valid records. **QA**

David E. Dvorin, Phoenix Solutions

Quick Tip

Enter Fractions or Decimals

When entering numbers containing fractions, such as the cost per share of a stock, you might like the option of typing the fraction rather than a decimal. Here's a program for a CostSh (cost per share) character field that converts a number containing a fraction (5/8, 3/4, 2/3, and so forth) to the equivalent two-decimal-place number:

```
> If @Instr(CostSh, "/") Then CostSh =
  @Round(@Left(CostSh, @Instr(CostSh, " ") -1) +
  @Mid(CostSh, @Instr(CostSh, "/") -1, 1) /
  @Mid(CostSh, @Instr(CostSh, "/") +1, 1),2)
```

If you enter "27 5/8," for example, Q&A will return "27.63" (27.625 rounded off to two decimal places). The program works by dividing the fraction's numerator by its denominator, adding the decimal result to the whole number, then rounding the number to two decimal places. You can make the program round the number to three decimal places simply by changing the final "2" to "3."

You could also add a Shares number field and TotCost money field to the database, and program Shares like this to calculate the total cost:

```
> TotCost = CostSh * Shares
```

QA

Tom Marcellus

The Quick Answer

Strung Out on @Instr

Jeff Nitka

I get a lot of mileage out of Q&A's @Instr (pronounced *at in-string*) function. In addition to using it to manipulate text values in a database, I use it to simplify code.

@Instr returns the starting position of the string you specify within a larger text value. For example, it can tell you the position of the first space in a field, a marker such as a "/" or "*" character, or where the "sing" in par*sing* begins. If field #100 contained *parsing*, #101 = @Instr(#100, "sing") would place "4" in field #101 because the "s" in *sing* begins at the fourth character position. If field #100 contained *Fourth of*July*, @Instr(#100, "*") would return "10" because the asterisk is at the 10th character position.

@Instr is typically used like this to extract text from a field. For example, a Contact field in a customer database might contain a whole name (the first name followed by a space, then the last name). To extract the last name and place it in a Last Name field, you could use @Instr this way:

```
Last Name = @Right( Contact, @Len(Contact)
            - @Instr(Contact, " ") )
```

Thus, if the Contact field contained *Jeff Nitka*, the Last Name field would contain *Nitka*, while the Contact field remained unchanged.

You can also use @Instr to tell you whether or not a specific string occurs within a field. In the following example, if *substandard* occurs anywhere in the Review Comments field, then the Bonus Factor field gets a goose egg:

```
If @Instr(Review Comments, "substandard")
Then Bonus Factor = "0"
Else Bonus Factor = ".2"
```

Simplifying programming

I use @Instr to reduce by half the number of @Xlookups in a looping routine. Here's the beginning of a sample loop:

```
<  If @XLu(@Fn, #1, "x#1", "Code") >= "0" and
      @XLu(@Fn, #1, "x#1", "Code") <= "9"
   Then ...
```

Using @Instr as follows, I can get away with just one @XLookup per loop:

```
<  If @Instr("0123456789",
      @XLu(@Fn, #1, "x#1", "Code") ) > 0
   then ...
```

@Instr as a verification tool

One of my databases (FORMULA.DTF) stores a list of ingredients and percentage levels for each product my company sells. Another database—RAWMAT.DTF for raw materials—tracks inventory. Each RAWMAT record contains a raw material, how much of it was consumed, and the product in which it was used.

I want to ensure that the user enters correct data, so I use an Ingredient field in FORMULA to store a concatenation of ingredients, each separated by a semicolon, like this:

```
;Ingredient1;Ingredient2;Ingredient3;Ingredient4;
```

This lets me use an @Instr statement like the following in RAWMAT's verification program:

```
> If @Instr( @XLu("FORMULA", Product, "Product",
  "Ingredient"), ";" + Raw Material + ";" ) = 0
  Then @Msg(Raw Material + " is not a standard
  ingredient in " + Product)
```

I surround the Raw Material value with semicolons to ensure accuracy. To see how this works, consider how the following statements would behave if the Raw Material field contained *Salt*:

```
If @Instr(";Calcium Salt;Rock Salt;Table Salt;",
   Raw Material) = 0

If @Instr(";Calcium Salt;Rock Salt;Table Salt;",
          ";" + Raw Material + ";") = 0
```

The first If fragment would return false because although *Salt* occurs in the target string value, it isn't an ingredient in the list (it isn't surrounded by semicolons). In contrast, the second If fragment would return true because *;Salt;* doesn't occur in the target string. The examples show how to construct text values as @Instr parameters that ensure accurate results.

Simplifying conditionals

You can use @Instr to simplify conditional expressions. Consider the following sample program:

```
Product Type = @XLu("ProdInfo", Product,
               "Product", "Type");
If Product Type = "Flammable Liquid"
   or Product Type = "Flammable Solid"
   or Product Type = "Combustible Liquid"
   or Product Type = "Poison"
Then Hazard = "Yes"
Else Hazard = "No"
```

Assuming Hazard were a Yes/No field, you could combine the two statements as follows:

```
Hazard = @Instr(";Flammable Liquid;Flammable Solid;" +
                 "Combustible Liquid;Poison;", ";" +
         @XLu("ProdInfo", Product, "Product", "Type") +
              ";" ) > 0
```

If you wrote the sample program with negative comparisons, like this:

```
If Product Type <> "Flammable Liquid"
   and Product Type <> "Flammable Solid"
   and Product Type <> "Combustible Liquid"
   and Product Type <> "Poison"
Then Hazard = "No"
Else Hazard = "Yes"
```

Then you'd need to modify @Instr like this:

```
If @Instr(";Flammable Liquid;Flammable Solid;" +
          "Combustible Liquid;Poison;",
          ";" + Product Type + ";" ) = 0
Then Hazard = "No"
Else Hazard = "Yes"
```

Although @Instr can be very useful, you must be careful with it. If any part of the first argument contains the second, the function will return a positive value—that is, a number greater than 0. Otherwise, it'll return 0. **QA**

Jeff Nitka works for a chemical manufacturer and develops Q&A applications part-time for Epoch Software, 908-874-3989. Jeff is the author of the Program Evaluator, a Q&A program debugging utility available from Marble Publications.

Quick Tip — Indent Subtotal and Subcount Labels to Improve Report Readability

When designing a report, if you specify a subtotal or subcount, Q&A still prints "Total" or "Count." Using a Column/Sort Spec like the following, you can replace the subcount "Count" label with "Subcount," and indent it a few spaces to improve overall readability:

```
Any field: 2,SC,C,H(5:Count)
State:  1,AS,SCL(  Subcount▸ ),CL(Total Count▸)
```

If you run this as a Totals-only report, you'll get a subcount of the number of records for each state, and a total count at the end, in a report like this:

```
State        Count
-----        -----
AK
             -----
   Subcount▸    37
AR
             -----
   Subcount▸    69
AZ
             -----
   Subcount▸   147
CA
             -----
   Subcount▸  1391
CO
             -----
   Subcount▸   165

. . .

============ =====
Total Count▸ 5000
```

You type the "▸" character by pressing Alt-F10, then Alt-16 on the numeric keypad. The technique works for total and subtotal labels as well. **QA**

Tom Marcellus

(Conceivably) Useful Date Calculations

Tom Marcellus

When you're at a loss for a date calculation solution, chances are you can can use Q&A's string and arithmetic functions to arrive at the date you need.

DOS WIN

WHILE reading a *PC Magazine*'s "Solutions" column the other day, my interest was piqued by a reader's programming question: "How do you reliably determine the number of days in a month? This is a problem that often stymies people." The reader went on to describe his FoxPro/Visual FoxPro program that performed the feat, then the column's editor added some commands to find the first and last dates of a month. Though I didn't have an immediate use for such a program, I wondered what it would take to do this in Q&A.

Figure 1 shows a sample date calculation database at the Format Spec. The field names and formatting codes are self-explanatory. **Listing 1** shows the on-exit program I came up with for the MyDate field.

(Keep in mind that Q&A stores dates in a YYYY/MM/DD format—1996/05/15 for May 15, 1996, for example—no matter the display format you select when you designed or redesign the database.)

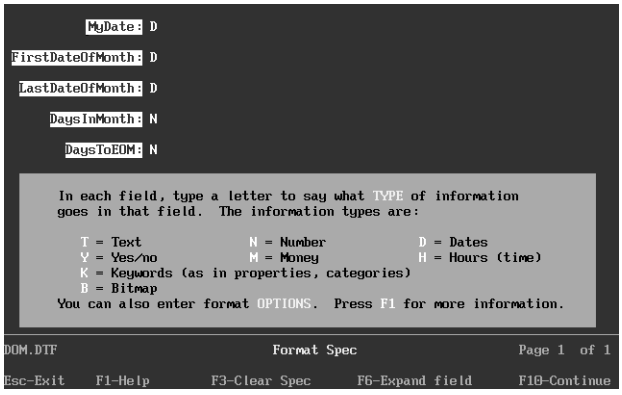

Figure 1. Fields and formatting for sample dates database.

Listing 1. A Q&A program that determines the number of days in the MyDate month, the number of days remaining in the month, and the dates of the first and last days of the month.

```
> FirstDateOfMonth = @Left(MyDate, 8) + "01";

  If @Month(MyDate) <> 12 Then
  LastDateOfMonth = @Todate(@Left(MyDate, 5) +
     @Str(@Month(MyDate) + 1) + "/01") -1
  Else
  LastDateOfMonth = @Left(MyDate, 8) + "31";

  DaysInMonth = @Dom(LastDateOfMonth);

  DaysToEOM = LastDateOfMonth - MyDate
```

The program returns the first date of MyDate's month by replacing the day number with "01." For all months but December, the last date of the month is found by taking the first date of the following month and subtracting one day from it. (For December, it simply makes the date December 31.) The number of days in the month is the day number of LastDateOfMonth, and MyDate is subtracted from LastDateOfMonth to get the number of days remaining in the month. (Add 1 to the last statement to count MyDate as a day.)

Figure 2 shows the results when the date is May 7, 1996. The program works in leap years, too. **QA**

Tom Marcellus is editor of *The Quick Answer* and author of *PC World Q&A Bible*, published by IDG Books. His QuickClick Calendar Plus—a time- and activity-tracking database for Q&A 5.0—along with his QuickClick RecordFinder for Q&A 5.0, are available from Marble Publications, publisher of *The Quick Answer*.

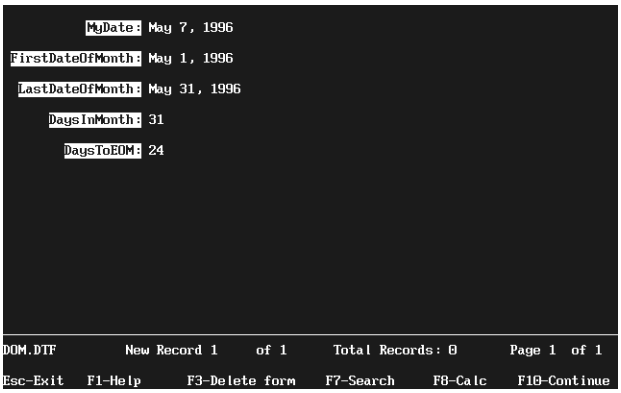

Figure 2. A sample record showing the results of the program's calculations.

Make Your Macros Loop the Loop

Jeff Nitka

A processing task can involve so many variables that you're loath to entrust it to a macro. Here's a nifty "divide and conquer" technique that loops a macro the exact number of times necessary to process each set of variable retrieval parameters.

When you can restrict the variable retrieval parameters involved in a processing task to one field, it's usually no problem automating that task with a macro. But if the macro must act on variable retrieval parameters in more than one field, or perform Mass Updates or other data manipulation operations, grouping the retrieval parameters can result in erroneous output or even data corruption.

I have three clients with processing tasks that fall into the latter category. Two run a complex multifield Mass Update on variable invoice line item records before printing a merge invoice. The other prints merge documents, each of which needs to include variable text in more than one field. In these cases, trying to specify all the retrieval restrictions in one place would be too complex and an invitation to error.

For these kinds of processing tasks, I came up with a technique that lets you specify multiple sets of multifield retrieval parameters at the outset, then run an auto-looping macro until all them have been processed.

How the technique works

With this technique, you add a record to a database named LIST.DTF, typing in the retrieval parameters for the macro to process. You save the first record, then add another with different parameters. You add as many records as you have distinct sets of parameters.

You then open a one-field database named GO.DTF that starts and stops the looping macro routine. GO.DTF's program checks LIST.DTF for the highest record number, then runs the *Process Info* macro to process it. If no LIST.DTF record exist, GO.DTF invokes the *Stop Processing* macro.

The trick is to design the *Process Info* macro to delete the LIST.DTF record just used, and pass control back to GO.DTF, so its program can continue the loop (rerun the macro) or terminate processing. To do this, the *Process Info* macro must perform these steps:

1. Exit GO.DTF, and open the database (or merge document) to be processed.

2. Call a saved Retrieve Spec (by pressing Alt-F8) that searches the database you're processing for the information that corresponds to the parameters in the largest LIST.DTF record. (I'll explain this in more detail later.)

3. Process the record(s) found, then delete from LIST.DTF the highest record number—the one just used in step 2.

4. Escape from LIST.DTF, and reopen GO.DTF in Add Data mode.

If there's another LIST.DTF record to process, GO.DTF reruns the *Process Info* macro. If LIST.DTF is empty (detected by the @Error function), then GO.DTF invokes the *Stop Processing* macro that exits GO.DTF and returns to the Q&A Main menu.

Setting up the databases

Let's create the two databases (LIST and GO), and run a demo using a macro that performs a Mass Update on some records that meet the retrieval parameters in two fields.

LIST.DTF contains three fields. Its first field, Entry, is formatted for numbers and is Speedy and read-only. Entry's program increments each record, like this:

```
< If @Add and Entry = ""
  Then Entry = @XLR(@Fn,999,"Entry","Entry") + 1;
  Cnext
```

The Product and Lot Number fields store retrieval parameters for the *Process Info* macro to act on. (You can use the Restrict Spec to place a "not empty" restriction these fields—it helps ensure accurate data retrieval and processing.)

GO.DTF contains a single text field named Question. Before you program it, though, record the two macros you'll need.

Recording the macros

Before recording the *Process Info* macro, you should create

and save any complex Update, Retrieve, or other Specs the macro will call.

Add a record to GO.DTF, and enter a dummy value (one character will do) in the Question field. Then, record the *Process Info* macro with the following steps:

1. Exit GO.DTF by pressing Esc and answering Yes. (You're not saving the record.)

2. Select Mass Update, enter the database path and filename to process (not LIST.DTF), and press Enter.

3. At the Retrieve Spec, press Alt-F8, and select the saved Spec that contains the following retrieval parameters:

   ```
   Product: = {@XLR("LIST",999,"Entry","Product")}

   Lot Number: = {@XLR("LIST",999,"Entry","Lot Number")}
   ```

   These expressions return the variable retrieval parameters from the LIST.DTF record with the highest Entry field number.

4. Press F10 for the Update Spec. Enter your Specs, or press Alt-F8 to select a saved Spec, then press F10 to run the update. When the update is finished, have the macro perform whatever additional steps are necessary to utilize the updated records, such as printing a merge document.

5. Select Search/Update for LIST.DTF, and press Enter.

6. Type "MAX" in the Entry field, and press F10.

7. Press F3, answer Yes, and press Enter to delete the record.

8. Exit LIST.DTF, select Add Data for GO.DTF, and press Enter.

Press Shift-F2, type "Process Info" as the macro name, then save it to your default macro file.

The second macro, *Stop Processing*, simply exits GO.DTF and returns to the Q&A Main menu:

```
<begdef><nokey><name>"Stop<sp>Processing"<vidoff>
<esc>Y<enter><esc><esc><enddef>
```

Programming GO.DTF

To control the looping process, enter the following program in GO.DTF's Question field:

```
< Question = @XLR("List",999,"Entry","Entry");
  If @Error
  Then @Macro("Stop Processing")
  Else @Macro("Process Info")
```

Press F10 to save the Program Spec, and you're done.

Try it out

You can now perform a test run. Add a few records to LIST.DTF. The first record should always contain "1" in the Entry field. If it doesn't, it means that a previous loop was terminated before LIST.DTF was emptied. In this case, you'll have to delete the existing records before you can begin a new run—unless you want to process them. Enter your retrieval variables in one record after another until the list is complete, then press Shift-F10 to save and exit.

Now, open GO.DTF in Add Data mode, and watch the looping process take off. If you have everything set up properly, the loop will run until LIST.DTF contains no more records. Good Luck! **QA**

Jeff Nitka works for a chemical manufacturer and develops Q&A applications part-time for Epoch Software, 908-874-3989. Jeff is the author of the Program Evaluator, a Q&A program debugging utility available from Marble Publications.

PO Box 9034
Gaithersburg, MD 20898-9034

Second-Class
Postage Paid at
Rockville, MD