T·H·E Quick Answer
January 1996
The independent monthly guide to Q&A expertise

# How to Merge Your Databases

What do you do when you have several databases with information you need to combine in a single database? Without proper preparation, you could wind up with a real mess on your hands. Make a clean job of it by following these "tricks of the trade."

*By David Dvorin*

DOS WIN

YOU might need to combine information from several databases into one. For example, you might have information about the same people in several databases that you'd like to have in a single database. Although Q&A can't perform the task directly, I'll show you a way to do it.

## Requirements

To merge several Q&A databases into one, the following three components are necessary.

- A database that combines all the information you want from all the related databases. I'll call this the *resulting database*. The databases whose information you want to merge into the resulting database I'll call the *source databases*.
- A field that provides a common link between the records in each database.
- A Mass Update Spec that performs the merge operation.

Let's look at each component in turn.

## The resulting database

The purpose of the resulting database is to combine the desired information from the records in the source databases.

You can create the resulting database by making a copy of the source database that contains the most information you want to merge with the resulting database. Use Q&A's Backup option (on the File / Utilities menu) to make the copy. This way, all the records are copied along with the form design, and it takes only a moment.

Next, redesign the copied database's form, adding the fields you want to merge from the other source databases.

## The linking field

Add one more field to each source database and the resulting database—a field to uniquely identify each

*Continues on page 3*

## CONTENTS

# README.1ST

MOMENTS after plucking his November *Quick Answer* from the mailbox, Jeff Nitka was on the phone reading me The Riot Act. The crux of his harangue was a tip of mine on page 9 of that issue ("Too Many Unhappy Returns") that contains—well, let's be charitable—a half-truth. To avoid a *Too many Returns* error message when Gosubs and Returns are executing, I advise to ". . . make the Return field read-only so you can't inadvertently move the cursor into it." But as Jeff pointed out, that won't cut it because although the content of a read-only Return field is uneditable from the keyboard, its program can still trigger, and you'll receive the error message when there's no pending Gosub to return to.

Jeff handles it by placing a small invisible field directly in front of the Return field, and programs it with a Navigation command to deflect the cursor. Fine, but even that isn't bullet proof. Suppose, for example, you press Down arrow from a field directly *above* the Return field?

The point is, you should barricade any field you want to keep the cursor away from except when your programming tells it to go there.

Did you know you can add gadgets to your Q&A 5.0 forms that work like the buttons, check boxes, dialog boxes, and drop-down lists you find in Windows? I demonstrated some of these at the September Q&A 5.0 Master's Seminar in Washington, D.C., and raised a few eyebrows. Find out how you, too, can take advantage of these labor-saving gadgets to simplify data entry.

Suppose you have information about the same people in more than one database, and you'd like to combine that information. The problem is Q&A doesn't know how, and you might not either. Merging databases can be tricky. The key, as **Dave Dvorin** shows, is in careful planning and preparation.

Go for those GOSUBs with **Jeff Nitka** in this month's Program Spec column.

Final call to send in your ad for *The Quick Answer*'s upcoming *Ad Data* Advertiser's Supplement. If you're a Q&A consultant, developer, user group leader, have a specialty database you'd like to distribute, or can supply Q&A-compatible add-on products, you must act by January 15 to make the February supplement. For details on preparing and placing your ad, see the November issue, or call or fax us at the numbers on this page.

**Tom Marcellus**
***Editor***


THE Quick Answer
*The independent monthly guide to Q&A expertise*

**Volume 7, Issue 1**

**Editor** Tom Marcellus
**Publisher** Michael Bell
**Copy Editor** Laurie Moloney

*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 pending 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: 503-465-8600.

# Merge Your Databases

*Continued from page 1*

record within each database and to identify the record as the same record across all the databases involved. This field will provide the link between the resulting database and the source databases.

If the pertinent databases contain people or companies, your best choice for the linking field might be a telephone number because it's unique, especially if it includes the area code. If the source databases don't all include a telephone number field, then you can use a combination of the first name, last name, and address.

There are a number of issues to keep in mind when deciding on a linking field. First, it must have the same structure in all the applicable databases. You can't use the telephone number as the linking field in one database, and a different linking field in another. Nor can you use a telephone number without an area code as the linking field in one database, and a telephone with the area code in another, though you could strip the area code from the telephone number in one database, then use the phone number without the area code as a linking field. (You could strip the area code by running a Mass Update on the database, with a programming statement that returns only the phone number. Or, the Update statement you use to merge the data into the resulting database could include a command to ignore the area code in the source database.) Also, no linking field value can be no more than 16 characters in length because this is the maximum number of characters that Q&A will treat as unique.

When you've decided on the linking field, add it as necessary to each source database and the resulting database. Then perform a Mass Update on each database to create the unique link field value. Suppose, for example, you decided to use the first six characters of a person's last name, plus the first three characters of the first name, plus the ZIP code to create a unique 14-character linking value. In each database, your Update Spec would include a program such as this:

```
First Name: #1
 Last Name: #2
  ZIP Code: #3
Link Value: #4 = @Left(#2,6) + @Left(#1,3) + #3
```

Using this statement, the linking value for David Dvorin in ZIP code 08876 would be DVORINDAV08876.

## Complete the resulting database

There's one more issue to resolve with the resulting database. You need to make sure all the records from the source databases are represented in the resulting database.

If there are records in any of the source databases not already in the resulting database, you can add them to the resulting database using Q&A's Copy command, copying selected records from one database to the other. You don't need to copy all the fields, just the linking field.

You should now have a resulting database that contains all the desired fields, and a record for each record in each source database. All the databases now have a common link. You can now use this link to run a Mass Update on the resulting database.

## The Mass Update Spec

For your a Mass Update Spec, you can use an XLookup command that references the newly created link fields in all the databases to conditionally copy information from the various source databases to the resulting database. Here's a sample Mass Update program for the resulting database's link field:

```
XLookup("Source", Resulting database link field name,
        "Source database link field name",
        "Source database merge field name",
        Resulting database merge field name)
```

Here's how this statement breaks down:

- *Source* is the name of one of the source databases.
- *Resulting database link field name* is the name you assigned to the linking field in the resulting database.
- *Source database link field name* is the name you assigned to the linking field in the source database.
- *Source database merge field name* is the name you assigned to a field you want copied from the source to the resulting database.
- *Resulting database merge field name* is the name you assigned to the field you want to receive the value from the source database.

For example, if one of your source databases is named CUSTOMER, your resulting database is named CLIENTS, your link field is named SLink and RLink in the source and resulting databases, respectively, and you're merging the field named Fax

in CUSTOMER to FaxNumber in CLIENTS, then your Mass Update program would look like this:

```
XLookup("CUSTOMER", RLink, "SLink", "Fax", FaxNumber)
```

You need to include an XLookup statement for each field being merged from each source database. For example, if you have three source databases and one, two, and three fields to merge from each one, respectively, then you'll need six Xlookup statements in the resulting database's link field.

(Advanced users note: Up to 23 merge fields from a single source database, along with up to 23 target fields in the resulting database can be incorporated into a single XLookup statement.)

After the Mass Update Spec is completed, save it so you can use it later during the merge process.

### Perform the merge

At this point, you should have all the necessary ingredients. You have a resulting database with all the fields you want. You have a common linking field in each source database and the resulting database. And you have a Mass Update Spec that will retrieve the data from the source database fields into the corresponding resulting database fields.

The merge process is nothing more than performing a Mass Update operation. Your Update Spec program will tell Q&A to search each source database, retrieve a field value from it, and place it in the corresponding field in the resulting database. When the update is complete, each record in the resulting database will contain the information from all the source databases, and your work will be done.

Merging databases this way is sometimes necessary. Although it can take a little work to properly set up, an approach like this ensures that the resulting database contains all the information you need from the source databases.

David Dvorin owns Phoenix Solutions of Hillsborough, New Jersey, which specializes in tailoring off-the-shelf software for broad range of business needs. 908-281-6272, Internet dvorin@bms.com.

---

QUICKTIP

## XUserselect Case-Sensitivity Caveat

DOS 5.0

XUserselect promises to be among the most popular programming commands in QA 5.0 for DOS because it lets you display a pick-list of values from an external database. This might be an external database you've set up exclusively for lookups. (The *Q&A 5.0 Application Programming Tools* (APT) manual even states ". . . external database, if it's not too large.") At other times, though, your pick-list of values will come from a "real" external database in which there might be duplicate entries in the indexed lookup field.

The APT manual also states that "No duplicate values are displayed." This is true, but only to a point. Unlike just about everywhere else in Q&A, XUserselect is *case-sensitive*, which means the same field entry with different capitalization isn't considered a duplicate value. Worse yet, if XUserselect finds an entry in uppercase letters and the next one in lowercase letters, it will list the same entry a third time if the next occurrence is again in uppercase letters.

It's easy to demonstrate this by retrieving a set of records with the same entry in the external key field, and changing just the initial letter of alternate records like this:

```
Record 1: Liverpool
Record 2: London
Record 3: london
Record 4: London
Record 5: london
Record 6: London
Record 7: LondoN
Record 8: london
```

The resulting XUserselect pick-list will include all these occurrences of London.

Alec Mulvey, Ascot, England

 

@HELP

*Edited by Dave Reid*

## Copy a Field's Value to Use Later

DOS 5.0

**I have a small problem with a Q&A application. After I add a new student record and save it, I run a macro that prints a mail-merge letter to that student. The macro places MAX in the Retrieve Spec Student ID field, and since the Student ID number in the database is auto-incremented, this technique always prints a letter addressed to the newly added student. Sometimes, I need to update an older record and print a merge-letter for that student, but my "MAX" macro doesn't work because the student's ID number isn't the highest in the database. How can I work around this?**

Ronnie Lichman, Washington, D.C.

Q&A 5.0's new Clipboard feature makes it possible to print a mail-merge letter from the displayed record no matter when that record was added to the database. You can redefine the macro so it includes the following steps:

1. Move to the Student ID field.
2. Press F11, Q&A 5.0's *Copy to Clipboard* command key.
3. Press Shift-F10 to save the record and exit.
4. Get your merge document, and prepare to print it.
5. At the Retrieve Spec, move to the Student ID field.
6. Press F12, the new *Paste from Clipboard* command key.
7. Press F10 as usual to print.

The revised macro will automatically print a merge document for any record onscreen. F11 and F12 can be used to copy a field's value to or from a record, a Retrieve Spec, or even a document.

*[Here's an old Q&A 3.0 trick you can use to get a "MAX"-based macro to select the last viewed record irrespective of when it was added to the database. Add a labelless field to the form in an out-of-the-way place. Name it something like MaxRecNo, and make it Read-only and Speedy. Next, program an on-record-exit statement in the database, like this:*

```
#100 = @XLookupR(@Fn, "999999", "MaxRecNo", "MaxRecNo")
       + 1
```

*This statement (field #100 is MaxRecNo) assigns the highest number to any record you save. This way, you can use your MAX retrieval parameter in the MaxRecNo field to select the last record viewed and saved.—Ed]*

## Calculate Elapsed Days Sans the Weekends

DOS WIN

**I use Q&A for DOS to track my department's projects. The database includes a Start Date field, a Completion Date field, and a Length (length of time) field. I've programmed the Length field like this:**

```
< Length = Completion Date - Start Date
```

**This statement correctly calculates the number of days it took to complete the project, but I'd like to improve on the calculation by omitting the weekends. Is there a way to have Q&A count just the weekdays?**

Stan Decowski, New Jersey

Sure is. The following program in the Length field calculates the number of weekdays between the Start Date and the Completion Date:

```
< Length = (Completion Date - Start Date)
  - (2 * @Int((Completion Date - Start Date)/7));

If @Instr("MonTueWedThuFri",
    @Left(@Dow$(Start Date),3)) >
   @Instr("MonTueWedThuFri",
    @Left(@Dow$(Completion Date),3))
Then Length = Length - 2
```

The program contains two statements. The first simply fills the Length field with the number of elapsed days, less two for every seven. The second conditionally subtracts an additional two days if the Start Date is later in the week than the Completion Date. For example, a project that begins on a Friday and ends on a Monday will work out to three days after the first statement's result because there were zero complete weeks. The second statement corrects this by subtracting 2, which yields the correct answer—one weekday. The program assumes that the Start Date and Completion Date are weekdays.

## Fix Creeping Envelope Addresses


**I use Q&A to print letters and envelopes for my business. Whenever I try to print more than one envelope at a time, the address immediately begins creeping further down the envelopes. I've verified that my document has some text below the address block, but the addresses are still on the move. How can I stop them?**

William Ruthrauff, Warwick, Rhode Island

Q&A can print to more than 400 different types of printers, and each of them has its own way of handling envelopes. By default, Q&A can easily communicate correctly with most of them, but a few require fine tuning.

Q&A provides a special setting to fine tune envelope printing. To adjust this setting, from the Q&A Main menu, select Utilities / Install Printer. Select the Printer (A through E) you use to print your envelopes, then select the Port, the Manufacturer, and finally the model. When you select the model, a confirmation box appears with information specific to your printer. Read it, then press F8 to set the Special Printer Options. Press F10 to proceed to the More Special Printer Options screen. You'll see an Envelope Height option. Because your addresses are creeping down the envelope, you should reduce your Envelope Height. Begin by reducing it by one, then print a few envelopes to see how it's working. If the address is still moving down, try reducing the setting even further. Eventually, you'll find the correct setting that works for your printer.

*[See also "Creeping Envelope Solutions" on page 16 of the August 1995 issue.—Ed.]*

Dave Reid is a Symantec senior support analyst providing second-level assistance to the technical support representatives. He's also the coauthor 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. 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.

---


## Macro-Driven Report Problem Solved

I had a baffling problem with a macro that printed a landscape report. The problem occurred after someone would change the number of spaces between columns from "1" to "Variable" in the Report Global Format Options. This would cause the report width to exceed the page width, and stop the macro at the *Report Too Wide* screen. I fixed this by adding commands to the beginning of the macro to reset the number of spaces between columns to 1:

```
rsfilename<enter>cf1<f10><esc><esc>
```

You can also optionally add the letter "S" (for *Split report across pages*) to a macro that prints a report. You insert it after the "N" that indicates "No" temporary changes, or after the final <f10> if the macro steps through some temporary changes. This way, if the report is too wide, the "S" tells Q&A to split it across multiple pages, and the macro won't be interrupted. (Later, you can determine which fields caused the width problem, and restrict them to a maximum length.) If the report isn't too wide, the "S" will begin to select Set Global Options on the Report menu after the report prints, so be sure to add an extra <esc> to back out of it.

Michael Schein, Reading, Pennsylvania

# Streamline Tasks with Custom Windows-like Gadgets

Sorry, graphical icons even in Q&A 5.0 are out of the question. But pick-lists, check boxes, dialog boxes, and buttons . . . Yep, —these we can do.

*By Tom Marcellus*

DOS 5.0

Q&A 5.0 brings powerful weapons to the war on data entry overload. And where it doesn't supply the weapons themselves, it gives you the tools to build them. Despite Q&A for DOS's non-graphical interface, you can create data entry gadgets that simulate some of Windows' best-loved features such as form buttons, radio buttons, check boxes, dialog boxes, and drop-down lists (see **Figure 1**). Designed to be used with your mouse, these gadgets can automate data entry like never before, cutting keystrokes and errors wholesale. I'll show you some situations where they can benefit you most, along with examples on how to design and program them.

## Choose your weapons

Figure 1 shows a Q&A 5.0 demonstration database with four Windows-like controls: form buttons, radio buttons, check boxes, and pick-list buttons. I'll refer to these gadgets collectively as *controls* or *control fields* because although they're fields, they don't store data—they control data entry and related tasks. They're designed to work in conjunction with a pointing device, so if you don't use a mouse with Q&A, they probably won't be of interest, at least until you see what they can do.

Briefly, here's what the controls shown in the **Figure 1** database do. I'll cover each type in detail:

- A partial pick-list for the SalesRep field is displayed in the upper right-hand corner of the screen. The SalesRep, State, CallBack, Terms, and Notes fields are all pick-list fields——the "▼" pick-list control character adjacent to them identifies them as such. The "▼" is actually a field. As you'll see, moving the cursor to an *empty* field displays its pick-list automatically. But you must click on the pick-list control character to display the list for a field that's already filled.

- A group of button controls appear in the lower left area of the form. These simulate buttons in Windows programs such as Q&A for Windows——that is, you click on them to perform tasks. The Figure 1 form includes buttons to write a *Letter*, print a mailing *Label*, create an *Invoice*, *Print* the record, create a *PostIt* note, switch to *Table* view, perform *Lookups*, and view the customer's *Trans*action history.

- Five check boxes, *Post*, *Copy*, *Mark*, *Xport*, and *Tag/Untag all*, appear in the lower left area of the form. Clicking on a check box fills it with an "X." This way, a program can make decisions and

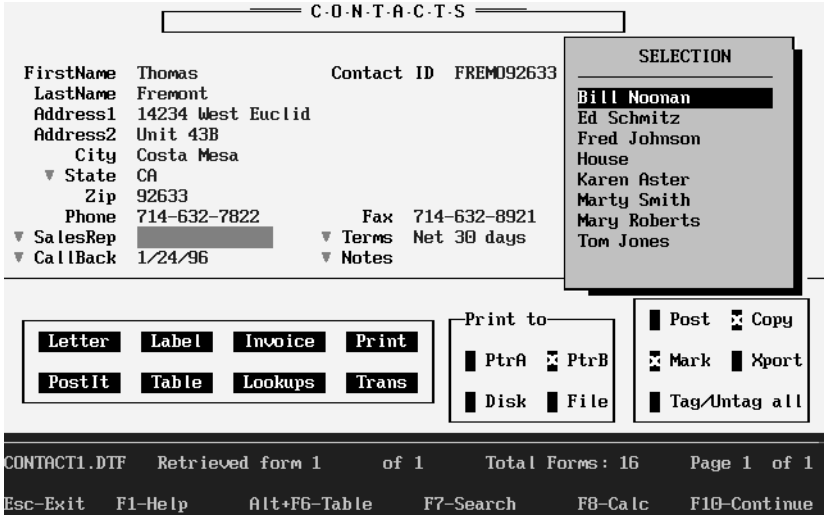


**Figure 1.** A sample Q&A 5.0 database with form buttons, radio buttons, check boxes, and pick-list controls to simplify data entry.

perform actions based on which boxes are checked. The *Tag/Untag all* check box lets you check or clear all the boxes in the group. In Figure 1, the *Copy* and *Mark* boxes are checked, indicating that actions relating to these fields are to be performed——when the record is saved, for example.

- A group of four radio buttons, *PtrA*, *PtrB*, *Disk*, and *File*, appear between the buttons and check boxes. They're called "radio" buttons because like a radio with push button station selectors, they're mutually exclusive. Though they look like check boxes, only one of the group of four can be checked. If you click on one to mark it, the other three are automatically unmarked. In this database, if you click on *PtrB* then click on the *Print* button, Q&A will print the record to whatever printer is installed as Printer B.

In addition to these controls, the Figure 1 form includes a *dialog box* control that's visible in **Figure 3**. Let's take a closer look at each type of control.

## The pick-list control

You use Q&A 5.0's list commands—Userselect and XUserselect—to display a list of field values, usually as you enter a field. Making your selections from pop-up lists rather than typing them minimizes keystrokes and helps avoid errors.

Let's focus on the SalesRep field in Figure 1. Notice the "▼" just to the left of it. It's actually a one-character field named SalesRepList. SalesRep stores the sales rep's name, while SalesRepList displays the pick-list of sales representatives.

Why two fields? When you're adding a new record, SalesRep will be empty, and you'll want the pick-list to display when you move to it. Once SalesRep is filled, however, you probably won't want the pick-list to display—rather, you'll want the *option* to display it if, for example, you're assigning a different sales rep. The SalesRepList pick-list control field gives you that option.

SalesRep's Navigation Spec program passes control to the SalesRepList field when SalesRep is empty:

```
< If SalesRep = "" then goto SalesRepList
```

This invokes SalesRepList's program, which displays the list:

```
< Userselect("Fred Johnson,Marty Smith,Mary Roberts,
Bill Noonan,EdSchmitz,Karen Aster,Tom Jones,House",
SalesRep);

If SalesRep <> "" then goto Terms
Else {@Msg("Please select a Sales Rep");
goto SalesRepList}
```

In this example, the Fax field that precedes the SalesRepList field includes an on-field-exit Goto that sends the cursor directly to SalesRep, bypassing SalesRepList. If SalesRep is empty, the pick-list appears. Once SalesRep is filled, the cursor is sent to the Terms field. If you try to escape from the pick-list when SalesRep is empty, the program sends the cursor back to SalesRepList, which redisplays the list. The pick-list doesn't pop-up automatically if the SalesRep field is filled. In that case, you must click on the "▼" to display the list and select your new sales rep.

The State, Terms, and CallBack fields—each has a pick-list—include similar pick-list control fields. Working in conjunction with form Navigation commands, the tandem fields let you move around a filled form without the pick-lists getting in the way. But you always have the option to display a list by clicking on the "▼" field.

You can fill your pick-list control fields with the "▼" character (or any character you choose) by making it an Initial Value, or by including it in an on-record-entry program like this:

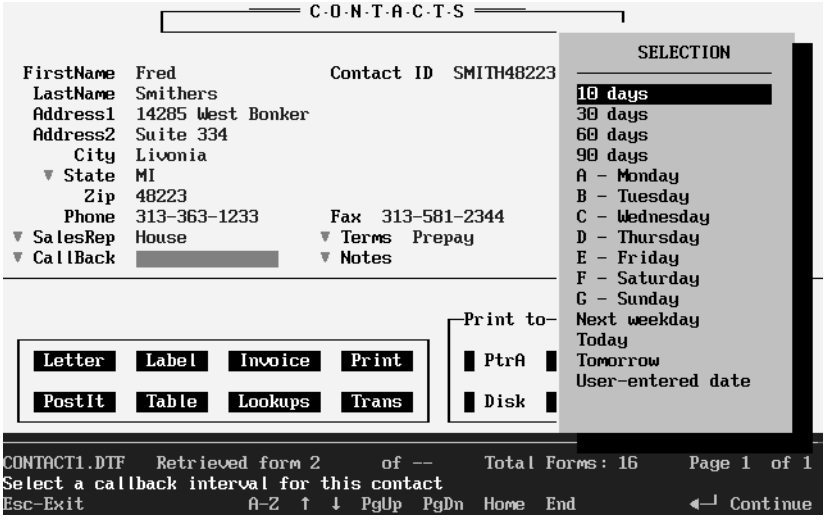


**Figure 2.** You can display a pick-list of callback intervals by moving to the empty Callback field or by clicking on its adjacent pick-list control character. See the sidebar, "Choosing a Callback Date" on page 9.

```
#5: StateList = "▼"; SalesRepList = "▼";
    TermsList = "▼"; CallbackList = "▼"
```

To create the "▼" character, press Alt-F10, then hold down the Alt key and type 31 on the numeric keypad.

The NotesOpen field ("▼") adjacent to the Notes field works a bit differently. It doesn't display a list, but opens the field editor for the Notes field. It's programmed like this:

```
< @Fedit; goto Notes
```

The CallBackList field ("▼") adjacent to the CallBack field also works differently than the other pick-list controls. Rather than display a list of values for CallBack, it displays a list of callback *time intervals*. When you select a callback interval, the program determines the future callback date, and places *that* in the CallBack field. See **Figure 2** and the sidebar, "Choosing a Callback Date."

Make your pick-list control fields single-character fields defined with the "<" and ">" field delimiters, and set their background color to the form's color. This way, all you'll see on the form is the "▼" character.

Now that you've seen how the pick-list controls work, let's look at the button controls.

## Choosing a Callback Date

The program for the CallBackList field displays a list of callback time intervals, converts your selection to the corresponding date, then places it in the CallBack field. Because a text field for interim calculations is needed and CallBack is a date field CallBackList serves double duty as the interim calculation field as well as the pick-list control field. Here's CallBackList's program:

```
<
@Msg("Select a callback interval for this contact");

Userselect("User-entered date,Today,Tomorrow,
Next weekday,10 days,30 days,60 days,90 days,
A - Monday,B - Tuesday,C - Wednesday,D - Thursday,
E - Friday,F - Saturday,G - Sunday", CallBackList);

If @Instr(CallBackList, "-") = 3 Then
{
CallBackList = @Del(CallBackList, 1, 4);
If @Dow$(@Date + 1) = CallBackList Then
  CallBack = @Date + 1
Else If @Dow$(@Date + 2) = CallBackList Then
  CallBack = @Date + 2
Else If @Dow$(@Date + 3) = CallBackList Then
  CallBack = @Date + 3
Else If @Dow$(@Date + 4) = CallBackList Then
  CallBack = @Date + 4
Else If @Dow$(@Date + 5) = CallBackList Then
  CallBack = @Date + 5
Else If @Dow$(@Date + 6) = CallBackList Then
  CallBack = @Date + 6
Else If @Dow$(@Date + 7) = CallBackList Then
  CallBack = @Date + 7
}

Else if CallBackList = "User-entered date" Then
{@Msg("Type in the Callback date and press Enter");
CallBackList = "▼"; Clear(CallBack); Goto CallBack}

Else if CallBackList = "Today" Then CallBack = @Date

Else if CallBackList = "Tomorrow" Then CallBack =
@Date + 1

Else if CallBackList = "Next weekday" Then
{
If @Dow$(@Date) = "Friday" Then CallBack = @Date + 3
Else If @Dow$(@Date) = "Saturday" Then CallBack =
@Date + 2
Else CallBack = @Date + 1
}

Else if CallBackList = "▼" then Chome
Else CallBack = @Date + @Tn(@Num(CallBackList));
CallBackList = "▼"; CHome
```

To display the days of the week (Monday through Sunday) on the Userselect list in chronological order, they're typed as *A–Monday*, *B–Tuesday*, *C–Wednesday*, and so forth. The program strips the prefix from the selected value.

The program's calculations are based on the current date. In other words, if you select *30 days*, the program adds 30 days to the current date and places the new date in the CallBack field. Similarly, if it's Thursday, and you select Thursday as the callback day, the program will place *next* Thursday's date in the CallBack field.

The "User-entered date" selection lets you manually enter the callback date, so you'll need a Navigation Spec program like this in the CallBack field:

```
< If CallBack = "" and CallBackList <> "▼" then
  Goto CallBackList
```

You'll also need a program like this in the CallBack field to restore the "▼" character (The "▼" character, by the way, is ASCII 16.):

```
> CallBackList = "▼"; CHome
```

Otherwise, the CallBack field works like the other pick-list fields. If you move to it and it's blank, the list displays automatically; otherwise, you must click on the "▼" to display the list and select a new callback interval.

## Button controls

Button fields, besides looking cool, can be fabulous time- and labor-savers. With a single mouse click, you can perform tasks that might otherwise take dozens of keystrokes. In a database, buttons can provide advantages over custom menus. For example, the task selections are all visible because their buttons are visible, and it takes just one click to invoke any task. However, you can always create buttons that do nothing but display custom menus.

Follow these steps to add a button field to your form:

1. Decide where to place the button field. You want to prevent the cursor from entering it unless you click on it. One option is to place the button at the bottom of the form, and program the fields just above it with on-field-exit Gotos to prevent the cursor from moving further down the form. You can place buttons along the right edge of the screen as long the data fields to their left include navigation Gotos to avoid them.

2. Decide on a "label" for the button. You'll place the label *in* the field, rather than in front of it.

3. When placing the button on the form, use the "<" and ">" characters to define it, and make it two characters wider than its label. This way, you'll have a space on either side of the label and the button will look better.

4. At the Format Spec, center-justify the field using the T,JC code (JM for justify middle in the Q&A 5.0 beta). This will center the label on the button.

5. At the Field Names Spec, name the field with its label plus a "Button" suffix. For example, if the button's label is *Letter*, name the field *LetterButton* to differentiate it from data fields.

6. At the Palette Spec, set the field's background color to contrast with the form's background color, then set a contrasting text color for the label.

7. Add the label text. You can set an Initial Value for the field, or use an on-record-entry program to assign the button labels when the form is displayed. For the eight Figure 1 buttons, such an on-record-entry program might look like this:

```
#10: LetterButton = "Letter"; LabelButton = "Label";
InvoiceButton = "Invoice"; PrintButton = "Print";
PostItButton = "PostIt"; TableButton = "Table";
LookupsButton = "Lookups"; TransButton = "Trans"
```

## Programming button fields

Following are some sample programs for the buttons shown in the Figure 1 form. They should give you ideas on how you can use buttons to reduce complex tasks to a single mouse click.

### *Letter*

The Letter button creates a name and address block (using the LetterButton field for temporary storage) from the information in the current record, then adds the date and a *Dear FirstName* salutation. The macro called at the end of the program then copies the block to the Clipboard, restores the LetterButton field, saves the record, switches to Write, and pastes the text into the new document in this format:

```
Fred Smithers
14205 West Bonker
Suite 334
Livonia, MI 48223              January 12, 1996

Dear Fred:
```

This way, all you have to do is type the text of your letter and closing, then print it. Here's the LetterButton program:

```
< If FirstName <> "" and LastName <> "" and
  Address1 <> "" and City <> "" and State <> "" and
  Zip <> "" Then
{
If @Askuser("Write a letter to",FirstName + " " +
  LastName +"?","") Then
{
If Address2 <> "" Then
    LetterButton = FirstName + " " + LastName + "
" + Address1 + "
" + Address2 + "
" + City + ", " + State + " " + Zip;

If Address2 = "" Then
    LetterButton = FirstName + " " + LastName + "
" + Address1 + "
" + City + ", " + State + " " + Zip;

LetterButton = LetterButton + @Text(15," ") +
@Month$(@Date) + " " + @Str(@Dom(@Date)) + ", " +
@Str(@Year(@Date)) + "
" + "
" + "Dear " + FirstName + ":";

@Macro("Letter to Current Contact")
}
Else Goto FirstName
}
```

The LetterButton program first makes sure the appropriate name and address fields are filled, then asks for confirmation that you want to write a letter to the current contact. (If you respond with "No," the program, in this case, moves the cursor to the FirstName field. If you'd like to return to the field you were in when you clicked on the button, see the sidebar, "Returning From a Control Field.")

At several places in the program, an opening double quote mark at the very end of a line is followed by a closing double quote mark at the

beginning of the next line. This signifies a carriage return. When typing a program, you can type a double quote mark, press Enter, then type another double quote mark to invoke a carriage return. In this case, the technique places the name, address, and so forth on separate lines, creating an actual address block.

The *Letter to Current Contact* macro "presses" F11 to copy the contents of the LabelButton field to the Clipboard, then opens the Field Editor, deletes the address block, and restores the Letter label to the button. Finally, it saves the record, displays the Write document screen, pastes the address block, then moves the cursor into position for you to begin typing the letter. Here's the macro:

```
<begdef><nokey><name>"Letter<sp>to<sp>current<sp>contact"
<vidoff><f11><f6><f3><end><end><end><f10><f6>Letter
<capsf10><esc>wt<f12><end><end><enter><enter><enddef>
```

### Label

The Label button prints a predesigned mailing label for the current record, then returns to the record. Here's the program:

```
< If @Askuser("Print a label for",FirstName +" "+
  LastName + "?","") Then
{
LabelButton = Contact ID;
@Macro("Print Label & Return")
}
Else Goto FirstName
```

The *Print Label & Return* macro copies the Contact ID (temporarily stored in the LabelButton field) to the Clipboard, then restores the button's label. It then saves the record and exits, selects the *Shipping Label* from the List of Mailing Labels, pastes the Contact ID into the Retrieve Spec, and prints the label. Finally, it returns to the original record by pasting the Contact ID (still in the Clipboard) into the Contact ID field at the Retrieve Spec. Here's the macro:

```
<begdef><nokey><name>"Print<sp>label<sp>&<sp>return"<vidoff>
<f11><f6><capsf4><f6>Label<capsf10><esc>wmShipping<sp>Label
<enter><f2><f10><tab><f12><f10><enter><esc><esc><esc>fs
<enter><tab><f12><f10><enddef>
```

### Invoice

The Invoice button starts a new invoice for the customer in the current record. It copies the Contact ID to the InvoiceButton field and runs the macro. Here's the program:

```
< If @Askuser("Create an invoice for",FirstName +" "+
  LastName + "?","") Then
{
InvoiceButton = Contact ID;
@Macro("Create Invoice")
}
Else Goto FirstName
```

The Create Invoice macro copies the Contact ID to the Clipboard, restores the Invoice label, opens the INVOICE database in Add Data, pastes the ID into the Customer ID field, then "presses" Enter to execute the INVOICE program that retrieves the customer address information into the new invoice. Here's the macro:

```
<begdef><nokey><name>"Create<sp>Invoice"<vidoff><f11>
<f6><capsf4><f6>Invoice<capsf10>a<capsf4><f10>invoice
<enter><f12><enter><enddef>
```

### PostIt

The PostIt button plays a little ditty before shelling out to another copy of Q&A to add a reminder note to the POST-IT database. When you exit the second copy of Q&A, you're returned to the same record. Here's PostIt's program:

```
< If @Askuser("Go to Post-it Notes?","","") Then
{
@Play("sound","196,75"); @Play("sound","256,75");
@Play("sound","329,75"); @Play("sound","392,150");
@Play("sound","329,75"); @Play("sound","392,150");

PostItButton = @Shell("D:\QA\QA.COM -alDEMO.ASC -m2");
PostItButton = "PostIt"; Chome
}
Else Goto FirstName
```

The @Shell command loads the macro file named DEMO.ASC and invokes its Alt-2 macro, which, in this case, opens the POST-IT database in Add Data mode.

### Table

The Table button switches from Form View to Table View. Its program contains just one command:

```
@Macro("Table View")
```

The Table View macro displays the Retrieve Spec, clears it, selects the Sort Spec named *Default*, then switches to the default Table View. Here's what the macro looks like:

```
<begdef><nokey><name>"Table<sp>View"<vidoff><f7><f3><f8>
<altf8>Default<enter><f10><altf6><capsf6><f10><enddef>
```

### Trans

The Trans button invokes a macro that displays a predesigned screen report of the current customer's transaction history (showing billings, payments, and the current outstanding balance, for example). The program copies the customer's Contact ID to the TransButton field, then runs the macro:

```
< If @Askuser("View transaction history for",FirstName
  +" "+ LastName + "?","") Then
{
TransButton = Contact ID;
@Macro("Display Trans History & Return")
}
Else Goto FirstName
```

The Display Trans History & Return macro copies the Contact ID to the Clipboard, restores the Trans label to the button, then saves and exits the record and runs the Transactions screen report in

TRANS.DTF (pasting the Contact ID into the report's Retrieve Spec). The macro pauses as the report is displayed and returns to the same record (again, by pasting the Customer ID into the Retrieve Spec) when the F10 resume key is pressed:

```
<begdef><nokey><name>"Display<sp>trans<sp>history<sp>&
<sp>return"<vidoff><f11><f6><capsf4><f6>Trans<capsf10>
<esc>rp<capsf4><f10>trans<enter>transactions<enter>y<dn>
<f12><f10><f10><f10><wait><f10><vidoff><esc><esc><esc>fs
<capsf4><f10>contacts<enter><tab><f12><f10><enddef>
```

### Lookups

The Lookups button displays a list of all the names in the database, allowing you to quickly switch to any record simply by pointing and clicking. This sample program assumes the database includes a Speedy, Read-only field named FullName that contains the contact's last name, a comma and space, then the first name. This way, the Lookups program can display the names in last name order:

```
< XUserselectR(@Fn, "FullName", "Contact ID",
             "A", "Z", LookupsButton);

If LookupsButton <> "Lookups" Then @Macro("Find
Contact")
Else Chome
```

The Find Contact macro copies the selected person's Contact ID to the Clipboard, restores the Lookups label, displays the Retrieve Spec, clears it, pastes the Contact ID, then "presses" F10 to bring up the selected record. Here's the macro:

```
<begdef><nokey><name>"Find<sp>Contact"<vidoff><f11>
<f6><capsf4>Lookups<f6><f7><f3><tab><f12><f10><enddef>
```

The range on this sample XUserselectR command is A through Z—all the names in the database. If your database contains hundreds of records, it won't make sense to display all of them—rather, you'll want to specify a range of names to lookup.

Unfortunately, Q&A doesn't include a counterpart to a Windows *dialog box* that you could pop-up to type in the range—but you can create one. See **Figure 3** and the sidebar, "How to Create a Custom Dialog Box."

## Check boxes

The five check boxes in the lower right corner of the Figure 1 form work differently than the button fields we've discussed. For one thing, checking them doesn't necessarily invoke an immediate action, but serves as a flag for an action to be initiated when, for example, your on-record-exit program is triggered.

In the Figure 1 form, the first four check boxes are named *PostCheckbox*, *CopyCheckbox*, *MarkCheckbox*, and *XportCheckbox*. Adopting a naming convention such as this identifies the field as a control field, so you won't inadvertently select it for some inappropriate task (such as using it as a merge field).

The program for the *MarkCheckbox* field is representative of the other check box field programs. It simply places an "X" in the field if it's blank, and clears the "X" if it's not, like this:

```
< If MarkCheckbox = "" then MarkCheckbox = "X"
  Else Clear(MarkCheckbox); Chome
```

The *Tag/Untag all* check box lets you check or uncheck all the check boxes with a single mouse click. Here's its program:

```
< If PostCheckbox = "X" or CopyCheckbox = "X" or
  MarkCheckbox = "X" or XportCheckbox = "X" Then
  Clear(PostCheckbox, CopyCheckbox,
        MarkCheckbox, XportCheckbox)
  Else {PostCheckbox = "X"; CopyCheckbox = "X";
  MarkCheckbox = "X"; XPortCheckbox = "X"}; Chome
```

You can have your on-record-exit program decide what to do based on which check boxes are marked. It can also leave some or all of the checked boxes checked for future reference, or clear them. In the case of the MarkCheckbox field, you might have your program initiate an action if the box is checked, but

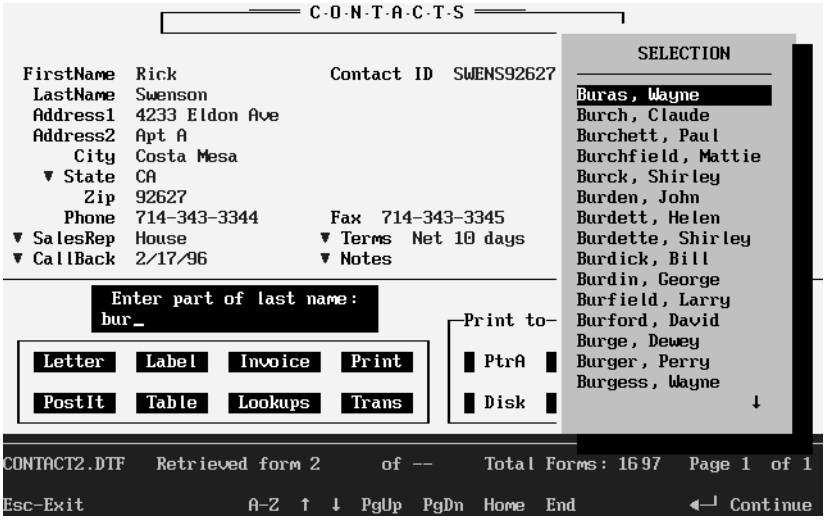


**Figure 3.** Your custom "dialog box" (above the buttons) prompts for the last name of the contact to lookup. The more characters you enter, the more you narrow down the lookup range.

clear it before initiating the action, like this:

```
If MarkCheckbox = "X" then {Clear(MarkCheckbox); action}
```

This way, the box won't be checked the next time you display the record.

To design a check box into your form, simply make it a one-character field using the "<" and ">" symbols. Place the label to the right of the field instead of to the left, assign an appropriate name to it at the Field Names Spec, and use the Palette Spec to give it contrasting background and text colors. Q&A won't regard "labels" that *follow* fields as genuine field labels, but you can assign the same color to the form's background text and the field labels. This way, our check box labels will appear in the same color as the actual field labels and therefore look like labels.

## Radio buttons

Radio buttons look like check boxes but act differently. They're mutually exclusive; that is, only one button in the group can be checked. The four radio buttons in the *Print to* section of the Figure 1 form, *PtrA*, *PtrB*, *Disk*, and *File*, are programmed to be mutually exclusive. Here's the program for the PtrBRadio field. The programs for the other three radio buttons follow the same logic:

```
< If PtrBRadio = "" Then
{
PtrBRadio = "X"; Clear(PtrARadio, DiskRadio, FileRadio)
}
Else PtrBRadio  = ""; Goto @(CurrentField)}
```

With this program, the PtrB box gets an "X" when you click on it (it's cleared if you click on it a second time), while the other three radio buttons are cleared. The *Goto @(CurrentField)* command returns you to the last data field you were in. See the sidebar, "Returning From a Control Field."

In this sample database, the *Print* button program requires that one of the radio buttons be checked before it will run a macro to print the record. If none of the radio buttons are checked, a message box appears prompting you to select the output device. Here's the *Print* button's program:

## How to Create a Custom Dialog Box

A dialog box typically appears and prompts for user input, accepts the input, stores it, then disappears. Though Q&A 5.0 doesn't include a built-in command to do this, you can create a gadget that looks and performs just like a dialog box.

Add two labeless fields to your form, one below the other. Make them about 30 characters in length, and use the "<" and ">" characters to define them. Name them *PromptText* and *PromptValue* at the Field Names Spec. At the Palette Spec, set their background and text colors to the same color as the form background. This way, they'll remain invisible until you choose to display them.

Using the *Lookups* button as an example, you can use the same *Find Contact* macro I described earlier, but change the program in the LookupsButton field to this:

```
< If PromptValue = "" Then
{
@Color(PromptText, 15, 0); @Color(PromptValue, 15,
0);
PromptText = "Enter part of last name:";
Goto PromptValue
}
Else
{
XUserselectR(@Fn, "FullName", "Contact ID",
    PromptValue, PromptValue, LookupsButton);
@Color(PromptText, 7, 7); @Color(PromptValue, 7, 7);
Clear(PromptText, PromptValue);
};
If LookupsButton <> "Lookups" Then
@Macro("Find Contact") Else Chome
```

Figure 3 shows what the screen looks like after you've clicked on the Lookups button and entered the first few characters of a last name—"bur," in this case.

The two fields create a serviceable dialog box that prompts for entry of the lookup range. The XUserselectR command uses PromptValue as the starting and ending ranges for the names to include on the pick-list. The more characters you enter in response to the prompt, the faster Q&A will display the names, and the shorter the resulting list will be.

The @Color commands set both fields to white text on black for the solid box effect. Once you've selected an item from the pick-list (or pressed Esc to abandon the search), the fields are cleared and returned to their original colors (white on white, in this case), making them disappear.

With a single dialog box such as this designed into your form, you can make it appear whenever you need to prompt for a temporary variable. You can have your program specify the PromptText, act appropriately on the PromptValue, then put the box away. Your program can even call an @Macro to copy the PromptValue to the Clipboard for later use.

```
<
If PtrARadio = "" and PtrBRadio = ""
and DiskRadio = "" and FileRadio = "" Then
{
@Msgbox(
"Choose an Output device","then click on Print","");
Chome
}
Else
{
If PtrARadio = "X" then @Macro("Print to PtrA")
Else if PtrBRadio = "X" then @Macro(Print to "PtrB")
Else if DiskRadio = "X" then @Macro("Print to Disk")
Else if FileRadio = "X" then @Macro("Print to File")
}
```

Like check boxes, you can use radio buttons to determine which of several actions a conditional on-record-exit program initiates, making execution dependent on one of the radio buttons being checked.

## Conclusion

Adding these Windows-like gadgets to your forms does more than boost productivity. They're visually appealing and provide a friendly interface in which to perform data entry and related tasks. They require additional fields and programming, so they might not be suitable for huge, heavily programmed databases. For most applications, though, a few pick-list controls and buttons can make working with Q&A that much more pleasant and productive.

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—is available from Marble Publications, publisher of *The Quick Answer*.

## Returning from a Control Field

You might be anywhere on a form—in an address field, for example—when you decide to mark a check box or radio button. Though you can always click on the same address field to return to it, you can program Q&A to save you the trouble. Here's how.

Add a one-character labeless field to the form in an out-of-the-way place. Name it LastField, make it Read-only, and color it to make it invisible. Next, add on-field-exit commands to your regular data fields similar to this one for the Address1 field:

```
> LastField = "Address1"; Goto Address2
```

When the cursor exits Address1, "Address1"—its field name—is copied to LastField. If you've pressed Tab or Enter, the Goto command moves you to Address2, as you'd expect. But if you click on a button or check box elsewhere on the form, and if that field includes *Goto @(CurrentField)* as its last command, Q&A will return you to the Address1 field automatically.

You can also use this technique with buttons whose programs first ask for confirmation before performing their task. This way, if you answer "No" to abort the task, you'll be returned to the field you were last in.

QUICKTIP

## Handy Record Tagging/Untagging

WIN

I use a field named Tag in my databases to tag the records I want to print, retrieve, update, or otherwise include in a procedure. To make it easier to tag and untag a record, I use the following program to toggle the Tag field between tagged and untagged state:

```
< #100: If #100 = "X" then #100 = "" Else
#100 = "X"
```

All I have to do is click on the field to tag (or untag) it. I use the same method to place an "X" or other character in Yes and No fields simply by clicking on them. It works great on questionnaire-type forms.

After I run the procedure that includes the tagged records, I use DAVE (the Do Anything Very Easily script recorder) and a Mass Update Spec to untag all the tagged records. In some applications, I have a button on the form that runs a Mass Update Spec named *Clear Tags*. In others, I have DAVE run the report then run *Clear Tags* to clear the tags. The Update Spec is #1 = "" in the Tag field, and the attached Retrieve includes an "X" in the Tag field. This way, Q&A can quickly find just the tagged records and untag them.

Bob Clark, Rohnert Park, California

*[You can use this technique in Q&A for DOS databases as well by substituting macros for the DAVE scripts.—Ed.]*

The Program Spec

By Jeff Nitka

# Go for the GOSUBs

WITH Q&A's GOSUB command, you can have your program GO to a SUBroutine field, execute its program, then return to the original field and execute the rest of the program. (Pages 83–84 of the Q&A *Application Programming Tools Manual* explain GOSUB basics.) GOSUB programming gives you these advantages:

1. It makes your programming more comprehensible.

2. It extends the amount of a field's programming by transferring some of it to other fields.

3. You avoid retyping blocks of code that perform the same task.

4. It boosts your program's efficiency.

I'll demonstrate the first and second advantages using my Program Evaluator (EVAL.DTF), a database that catches Q&A programming errors.

EVAL's Status field's program does most of the work. Without GOSUBs, Status would require some 600 lines of code. With 30 GOSUBs, I kept it to 339. Moreover, because some of Status' code is rarely invoked, I usually don't want to see it when I'm reviewing or editing the program. So, I simply copied that code to another field and tacked a RETURN statement onto the end of it. Here's what part of the program looked like before utilizing the GOSUB:

```
<#3: If Loc <= Len and (Status <> "Error") then {
     If Lead Token = "LEFTPAREN" then {
     If @Lt(Func,@in(Func,";")-1) = "@D("
     then { Str = ""; #50 = Loc; #51 = Pos; GoSub Ls;
            if Error > 0 then goto #3 } }
```

The code following the second *Then* is rarely called, so I "GOSUB" it to another field (#4), like this:

```
<#3: If Loc <= Len and (#3 <> "Error") then {
     If Lead Token = "LEFTPAREN" then { GoSub #4;
     If Error > 0 then goto #3 } }

<#4: If @Lt(Func,@in(Func,";")-1) = "@D(" then
     { Str = ""; #50 = Loc; #51 = Pos; GoSub Ls };
RETURN
```

When your program calls a GOSUB, the target field must contain a RETURN command to return program execution to the originating field. (You can also use the STOP command to halt GOSUB execution. STOP has the effect of pressing Esc, which stops all program execution).

To see how the third and fourth advantages work, consider a line-item invoice database that calculates a discount based on several factors for each item. The program retrieves pertinent information on an item, stores the data in temporary variable fields, computes the discount, then repeats the process for each additional item in the invoice. Here's the program without a GOSUB. ("Discount = .." refers to the code that calculates the discount information.):

```
< If #1 <> "" then { XLu("ExtFile",#1,"Prod","x#2",
  Var1,"x#3",Var2,"x#4",Var3); Discount = .. };
  If #2 <> "" then { XLu("ExtFile",#2,"Prod","x#2",
  Var1,"x#3",Var2,"x#4",Var3); Discount = .. };
  If #3 <> "" then { XLu("ExtFile",#3,"Prod","x#2",
  Var1,"x#3",Var2,"x#4",Var3); Discount = .. };
```

Notice the similarity of the code following each *Then* command. Instead of retyping or copying these code blocks, I use another field (named PTR, for "pointer") that references fields #1, #2, and #3. This way, I can use GOSUBs to economize on the programming:

```
< If #1 <> "" then { PTR = 1; GOSUB PTR };
  If #2 <> "" then { PTR = 2; GOSUB PTR };
  If #3 <> "" then { PTR = 3; GOSUB PTR };
```

Here's PTR's code:

```
< XLu("ExtFile", @( PTR ),"Prod","x#2", Var1,"x#3",Var2,
  "x#4",Var3); Discount = .. ; RETURN
```

The more extensive the code to calculate Discount, the more the technique economizes and increases program efficiency.

## Additional considerations

A GOSUB target field is unique in that the only way you can pass control to yet another field is by calling another GOSUB. In other words, you generally can't use navigation commands such as CNext, Goto, CHome, and so forth—they'll earn you a *RETURN statement missing* error message.

Moreover, you have to design the form so you can't accidentally enter such fields; you'll receive a *Too many RETURNs* error message, which means that a RETURN command was encountered with no pending GOSUB to return to.

For example, you can have a Yes/No field that contains Yes if a GOSUB is in process, and structure your program like this:

```
< if #1 <> ""
  then { Active = "Yes"; GOSUB PTR; Active = "No" };
```

The target field (PTR, in this case) would look like this:

```
< if Active then { blah; blah; blah; RETURN }
```

Should the cursor enter PTR, Q&A won't execute its program unless Active is set to Yes.

You can also place all the fields called by GOSUBs on one page and use a navigation command in the last field on the previous page:

```
Last Field: > Goto Last Field
```

To make the page even more inaccessible, you can place an extra field on it's first line, and program it like this:

```
Extra Field: < Goto Last Field
```

This way, pressing PgDn from the preceding page will have no effect.

Jeff Nitka holds a Bachelor of Science degree in mathematics and computer science. He develops Q&A applications part-time for Epoch Software, 908-874-3989. Jeff is the author of the Q&A Program Evaluator, a program debugging utility available from Marble Publications, Inc.

QUICKTIP

## Convert Text Values to Money Format

DOS WIN

Your routine to convert text values to money format (October 1995 issue, page 14) can be programmed more efficiently, like this:

```
> If Amount <> "" Then
{
Amount = @Str(@Tonumber(Amount) +
0.001);
Amount = "$" + @Left(Amount,
@Len(Amount) - 1)
}
```

You can adapt this program to display as many decimal places as you need. Simply change the number of zeroes to the right of the decimal point in the number added. In this case, the .001 adds two decimal places.

Jim Pogany, Oakville, Ontario, Canada

PO Box 9034
Gaithersburg, MD 20898-9034

Second-Class Postage Pending at Rockville, MD