Simplify Code and Subcode Lookups

IZZY GINDI

For accounting, inventory, and other systems that track things by codes and subcodes, you need a different kind of lookup technique.

MOST accounting systems are based on a chart of accounts comprised of codes and subcodes and their descriptions. *Office expenses*, to take one category, might have a code of 1100. Under it, there might be subcode descriptions for rent, utilities, telephone expense, and so on, with corresponding subcodes ranging from 1101 to 1199. (See the table on page 3.)

When I set out to create an accounting system for my business, I had my chart of accounts in a CODES.DTF database, but ran into trouble when looking up the codes and subcodes. When I knew the code and subcode, I wanted to enter them and have Q&A retrieve their corresponding descriptions. But when I couldn't recall the right code or subcode, I needed to display a pick-list of code *descriptions* from CODES.DTF, and have my selection retrieve the corresponding code or subcode. I didn't want to be dependent on a printed list to find the right code or subcode. Eventually, I found a solution, and Tom Marcellus helped me polish it up. To illustrate it, I'll use a payables database, but you can adapt the technique to any bookkeeping or other system that tracks things by codes and subcodes.

The databases

CODES.DTF (see **Figure 1** on page 3) is the chart of accounts database where the codes, subcodes and their descriptions are stored. PAYABLES.DTF (See **Figure 2**) is where payables are entered and assigned an account code and subcode. (Both databases are available on disk from Marble Publications. See page 4.)

Here's how they work. After entering the details about a payable (the vendor, the amount, and so forth), you enter the code, and the description field is automatically filled. If you can't remember the correct code, a convenient list of code descriptions from CODES.DTF appears. You select one, and the program retrieves the code and its description into the corresponding fields.

You then enter the subcode. If you can't remember it, a list of subcode descriptions from CODES.DTF appears. You choose one, and the program retrieves the subcode and its description into the corresponding fields.

You deal with the "parent" code first because you don't want a selection list with hundreds of subcode descriptions. The code enables you to "drill down" to a list that contains just the subcode descriptions whose subcodes fall within the parent code range.

Programming validates all codes and subcodes, so you can't enter a code that doesn't exist, or a subcode that isn't valid for its parent code.

The process is fast and reliable. Anyone who knows bookkeeping basics (such as which categories to assign different kinds of expenses) will get the hang of it in no time, even if they have little or no data entry experience. And all your payables will be accurately coded so you can run crosstabs or other reports to analyze expenses by category.

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

August 1997
Volume 8, Number 8

MARBLE PUBLICATIONS

Restrictions and assumptions

Codes and subcodes are always related, but not always sequential. (There might be gaps in the sequences as new codes and subcodes are added and old ones deleted.) The technique I use accommodates this, and assumes the following about the CODES.DTF records:

- Each CODES.DTF record includes a code, code description, subcode, and subcode description.
- All codes are numeric, and can't exceed 16 digits.

Continues on page 3

See Y'all in Savannah!

TOM MARCELLUS

FOR many who came to last year's National Q&A User Group Bash and *Quick Answer* Masters Seminar, the most productive part of the weekend proved to be the workshops. We picked a variety of topics—design, programming, reports, import/export, mail-merge, and so on—put someone with expertise on the topic at each table, and urged everyone to "circulate" until all their Q&A questions were answered. For hours, the large hall was a veritable hub of activity as attendees flitted from table to table getting expert help with their Q&A problems.

I led the workshop on data import/export, and I can't recall ever seeing so many satisfied customers. Some of the problems were tough, and I didn't have all the answers. But among the Q&Aers at my table, there was enough combined know-how to ensure that everyone went home with valuable insights, if not the exact solutions they were after.

So, whether you come for a fun, relaxing weekend with fellow Q&A enthusiasts, to watch the experts make Q&A perform amazing feats on the Big Screen, or to get expert help with your Q&A problems, the place to be October 24-26 is Savannah at the Holiday Inn, Midtown. See the insert in last month's issue (or turn to page 9) for information on how to register.

If I had to choose one *indispensable* Q&A feature, I'd pick this month's theme—lookups. Canadian **Izzy Gindi**, in his first *Quick Answer* article, reveals the innovative lookup techniques at the heart of his Q&A accounting system. **Peter Venuto** shows you a clever approach that uses lookups to control which macros can be run from a database. And I'll show you how to create lookups that let you instantly switch between record ranges without typing a character. Of course, all these solutions can be adapted to meet your own unique requirements.

South African **David Flaks**, also a first-time contributor, describes the technical problems he tackled to simplify a client's payroll tasks. And **Erika Yoxall** continues her tutorial on Q&A macros. T.J Shuflin is off this month. He'll be back next month with the @Help column.

The Quick Answer™

The Independent Guide to Q&A Expertise

Editor Tom Marcellus
Publisher Michael Bell

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.

Periodicals 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.

For Q&A technical support, call Symantec 503-465-8600.

Quick Tip

Too Quick for Q&A?

Random problems can occur with Q&A 5.0 on the newer Pentium Pro and Pentium II PCs. Some users have reported receiving *divide overflow* or *divide by zero* errors when Q&A 5.0 loads. According to Q&A 5.0's developer (Andreas Goebel, PFP Software, Germany), it has to do with the way Q&A checks for the type of processor present.

One solution is to slow the processor down a bit—and you can, using a utility originally designed to make PC games playable (run slower) on faster machines. You can download such a utility free of charge from the National Q&A Users Group's Web site at www.qaug.com, or from Micron Technology's BBS at 1-800-270-1207. The file to download, *Slowdown.exe*, is a 20K self-extracting zip file that uncompresses into *Moslow.com*, and *Moslow.doc*, an ASCII file containing instructions.

You use Moslow.com to retard the processor for "problem programs" only—it doesn't affect your PC's overall performance or system clock. Instead of starting Q&A with QA.COM alone, you use a DOS command like the following (optionally in a batch file named QA.BAT) to slow the processor to 95%:

```
moslow /950 qa.com
```

This seems to do the trick on Pentium Pro 200Mhz processors, for a net speed of about 190Mhz. We tried it on a Pentium Pro 266Mhz PC, and had to go to 70%—for a net speed of 186Mhz—to get Q&A to run. When you exit Q&A, the processor returns to its normal speed.

Gordon Meigs, PCTA, 215-598-8440, gmeigs@compuserve.com

Code Lookups . . . *continued from page 1*

- All subcodes are *unique*, numeric, and can't exceed 16 digits.
- All subcode descriptions are unique through the first 16 characters.
- Every subcode is numerically greater than its parent code, and less than the next higher parent code.
- Subcodes need not be sequential (there can be gaps), but the *range* of subcodes must be the same for all codes. In other words, the subcode range can be 25, 50, or 100 or more (you choose the range), but that range must be consistent across all parent codes.

A few sample accounting codes and subcodes with a range of 100 might look like this:

Code	Code Description	Subcode	Subcode Description
1000	Equipment Purchased *Subcode range* *1001 to 1099*	1072 1024 1039	Telephone Device Computer Hardware Fax Machine
1100	Office *Subcode range* *1101 to 1199*	1161 1109 1184	Rent Telephone Utilities

CODES.DTF contains four fields—**Code**, **Code Description**, **Subcode**, and **Subcode Description**. PAYABLES.DTF contains the same four fields.

CODES.DTF requires no programming, but its Code and Code Description fields must be Speedy, and its Subcode and Subcode Description fields must be Speedy Unique (SU). The Code and Subcode fields in both databases must be formatted N for Numbers. In PAYABLES.DTF, Code Description and Subcode Description should be read-only.

PAYABLES.DTF requires one additional read-only field named **List**. To make the programming work for the code range you're using, go to PAYABLES.DTF's Lookup Table, type *Subcode Range* in the Key column, and in the corresponding column 1, enter the range of subcodes you've established for your chart of accounts, such as 25, 50, or 100. If the Lookup Table contains *50*, it indicates your sequence of codes in CODES.DTF is on the order of 900, 950, 1000, 1050, or the like. If the Lookup Table contains *100*, it means the sequence of codes is on the order of 800, 900, 1000, 1100, and so on.

Programming

Listing 1 shows the programs for the Code, Subcode, and List fields in PAYABLES.DTF. Following is a description of what these programs do.

Code field program

If the Code field is left empty, a list of code descriptions appears courtesy of the @XUserselect command. (Although CODES.DTF contains many records with the same code description, the @XUserselect list will include only one of each *unique* description.) When you select one, the program fills the Code and Code Description fields, then moves to the Subcode field.

If you manually enter the code, the program checks to see if it's valid. If it isn't, an error message appears, and the cursor returns to the Code field. Otherwise, the code's description is returned to the Code Description field, and the cursor moves to the Subcode field.

Subcode field program

To illustrate how the Subcode program works, let's assume the code is 1100 (Office Expenses), and the subcode range is 100. In this case, the highest possible subcode is 1199. If a subcode hasn't been entered, the program sets the Subcode field to 1199 (the code, plus the Lookup Table entry minus 1), and executes an @XLookupR.

If a CODES.DTF record with an 1199 subcode is found, its subcode description is returned to the Subcode Description field. If no matching record is found, the

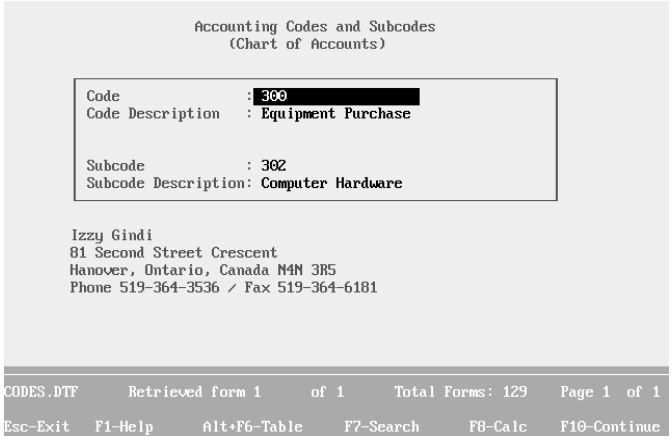


Figure 1. A typical CODES.DTF record. The Code and Code Description fields are Speedy. The other two fields are Speedy/Unique

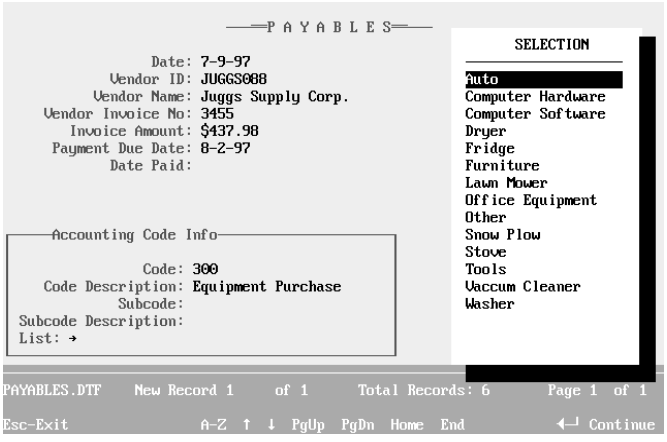


Figure 2. A PAYABLES.DTF record with a list of subcode descriptions displayed. The selection will return the description and subcode.

@XLookupR returns the next lowest Subcode (in this case 1184 for *Utilities*) to the Subcode field. This 1184 is now used as a matching field to retrieve the subcode description (*Utilities*) from CODES.DTF. This subcode description is then copied to the List field, and the cursor moves to the List field where its program takes over and assembles the complete list of subcode descriptions.

The second part of Subcode's program accommodates manual subcode entry. If the subcode isn't within the code's range, an error message appears, and the cursor returns to the Subcode field. If the subcode is within the range, the corresponding subcode description is retrieved. If the lookup fails (returns no subcode description), the error message appears, the offending subcode is cleared, and the cursor returns to the Subcode field. Only a legitimate subcode can run this gauntlet and return a corresponding subcode description.

List field program

Though every CODES.DTF record contains a subcode and subcode description, the objective is to retrieve only the subcode descriptions that relate to the parent code. In other words, we want to create a list of related subcode descriptions in the List field (*Utilities,Rent,Telephone*), then use @Userselect to display the completed list

To do this, a loop routine is employed to retrieve the subcodes. Continuing with our scenario, by decreasing the last valid subcode by 1, then executing another @XLookupR, the next subcode retrieved (according to the table on page 3) will be 1161. That subcode's description (*Rent*) is then retrieved, and added to the List field, separated from *Utilities* by a comma.

The loop continues as long as the subcode is greater than the code. Each subcode description is added to the List field until the loop terminates and the list is displayed in alphabetical order. When you select the subcode description from the list, its corresponding subcode is retrieved, and the List field is cleared.

Alternative Programming

If the quantity of subcodes is more or less the same for each parent code, and the subcodes are mainly sequential (with few or no gaps), you can make the subcode part of the routine run faster by changing a few things:

1. Set the Subcode field to the Code field plus 1.
2. Retrieve the corresponding subcode description and add it to the List field.
3. Use a loop to increase the Subcode field by 1, then retrieve its corresponding subcode description.
4. Terminate the loop after the highest possible subcode has been looked up.

Listing 1. Programming for the PAYABLES.DTF fields.

Navigation Spec:

Code field

```
< @Msg("Type the Code, or press Enter for list")
```

Subcode field

```
< @Msg("Type the Subcode, or press Enter for list")
```

Program Spec:

Code field

```
> If Code = "" then { Code Description =
@XUserselect ("Codes", "Code Description") ;
Code = @Xlookup("Codes", Code Description, "Code
Description", "Code" ) } ;

If Code <> "" then { Code Description =
@Xlookup ("Codes", Code, "Code" , "Code Description") };
If @Error then { @Msgbox (Code,"is an invalid
code.","Please try again.");
Clear (Code) ; Goto Code} Else Goto Subcode
```

Subcode field

```
> If Subcode = "" then
{ Subcode = Code + (@Lookup("Subcode Range", 1) -1 );
Subcode =
@XLookupR("Codes", Subcode, "Subcode", "Subcode");
Subcode Description =
@XLookup("Codes", Subcode, "Subcode", "Subcode
Description"); List = Subcode Description ; Goto List};

If Subcode < Code or Subcode >= Code + @Lookup("Subcode
Range", 1) then
{ @Msgbox(Subcode, "is an invalid Subcode.","Please try
again."); Clear(Subcode); Goto Subcode}

Else Subcode Description =
@Xlookup("Codes", Subcode, "Subcode", "Subcode
Description");
If @Error then {
@Msgbox(Subcode, "is an invalid Subcode.","Please try
again.");
Clear(Subcode); Goto Subcode} Else Chome
```

List field

```
< Subcode = Subcode - 1 ;
If Subcode > Code and Subcode < Code + @Lookup("Subcode
Range",1) then
{ Subcode = @XLookupR("Codes", Subcode, "Subcode",
"Subcode");
@Msg("") ; Subcode Description =
@XLookup("Codes", Subcode, "Subcode", "Subcode
Description");
List = List + "," + Subcode Description; Goto List };

If Subcode <= Code then { Clear(Subcode, Subcode
Description);
Subcode Description = @Userselect(List);
Subcode =
@XLookup("Codes", Subcode Description, "Subcode
Description", "Subcode");
Clear(List) ; Chome }
```

Izzy Gindi lives in Hanover, Ontario, Canada, and creates Q&A applications for himself and clients. 519-364-3536, fax 519-364-6181.

The two databases featured in this article (CODES.DTF and PAYABLES.DTF) are available ready-to-use from Marble Publications for $24 postpaid. (Specify disk No. **IG0897**.) To order by chargecard, call 800-780-5474 or fax to 301-424-1658. Or mail check to Marble at the address on page 2.

The Quick Answer

Jump-Search that Database

TOM MARCELLUS

Instantly switch between different ranges of records without typing a character.

SOMETIMES when I'm bored I fool around with Symantec's popular contact manager, ACT! for Windows. One thing I like about ACT! is that I can "jump" from any company record to companies beginning with the next or previous letter of the alphabet. I wanted this feature in my Q&A 5.0 contacts database, so I decided to work out the details. I call it my *alpha jump* technique, and here's how it works:

- There are three small "button" fields on the form—one that jumps to the next alphabetical range of records, another that jumps to the previous range, and a third that lets you select a specific range from A to Z.
- When you jump to the next, previous, or a specific A to Z range of records, they're displayed in sorted order. So, if you're on the Loonie Auto Parts record, and you click the *next* range, you get the first "M" record—Maaco Transmissions. If no "M" records exist, you get the "N" records—the next available range.
- Similarly, if you're on the Jiffy Products record, and you click the *previous* range, the first "I" record (IBM Corp., for example) appears. If there are no "I" records, the first "H" record appears.

You can add the alpha jump feature to any Q&A 5.0 database, and make it work with company names, last names, product names, or any Speedy field you want to display in alphabetical order. No matter what record is onscreen, alpha jumping gives you a typing-free shortcut to the range of records you're after. It's particularly helpful if you don't know how the company or last name is spelled, just the letter it begins with. It's also useful when the database contains records with duplicate search values (common last names like Smith and Jones, for example), and you can't use XUserselect(@Fn...) because only one such name would appear on the list.

(The pre-programmed ready-to-use database in Figure 1 is available from Marble Publications. See p. 7.)

Adding and customizing the fields

Start by adding three text fields to the database. Make each one two characters wide, and place them one right after the other without field labels, as shown in Figure 1. Next, go to the Field Names Spec (File / Design a File / Program a File / Set Field Names), and name them **JumpNext**, **JumpSelect**, and **JumpPrev** from left to right.

Finally, go to the Palette Spec (File / Design a File / Customize a File / Change Palette). Set their background color to contrast with the form's background color, and set a contrasting text color. This way, they'll look like three rectangular buttons on the form.

(You'll want to prevent the cursor from entering the Jump fields unless you deliberately click on one. The easiest way to do this is to place a *>Goto fieldname* navigation command in the field previous to the JumpNext field. Replace *fieldname* with the name or field ID of the next regular field to go to.)

Programming

You'll need an on-record-entry program to set the "labels" for the three button fields ("**>>**" for JumpNext, "**< >**" for JumpSelect, and "**<<**" for JumpPrev). If the database already contains an on-record-entry program (press F8 at the Program Spec to find out), add the following three statements to the end of that program. Otherwise, assign a new field ID number (#200 in the example) to any field that doesn't contain a program, type the following program in that field, press F8, type *200* on the *On entry field id* line, and press F10:

```
#200:    JumpNext   =  ">>";
         JumpSelect =  "<>";
         JumpPrev   =  "<<"
```

Following are the programs and descriptions for the three Jump fields. In these examples, the search field is named **Company**. If you're using a field named Last Name, replace all occurrences of *Company* with *Last Name.* Be sure the field is Speedy:

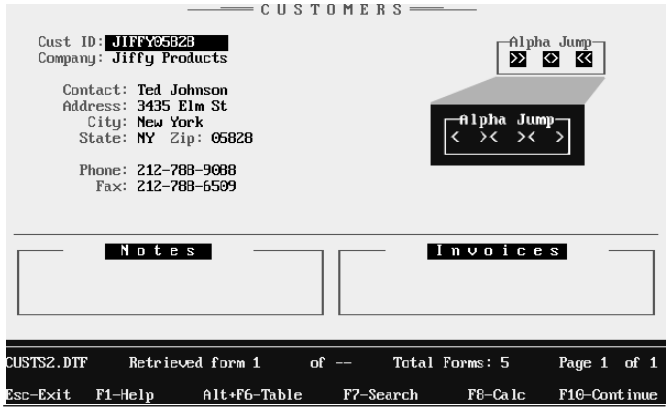


Figure 1. A sample database record showing how the three Jump fields look in the top right corner of the form. (The inset shows how they look at the design screen.)

1. JumpNext (>>) is the first of the three button fields. When you click on it, the program plays a "click" sound, takes the letter that's one higher than the first letter of the company name, then adds "ZZ" to it. (If you're on the "Jiffy Products" record, "J" is the first letter, so the program returns "KZZ.")

If that next higher letter is greater than 90 (the ASCII decimal value for the capital "Z"), the field is blanked, and an @Msg message tells you you're at the end of the alphabet. Otherwise, an @XLookupR range lookup is performed to determine if a record exists starting with that next higher letter.

If a qualifying record is found, the program adds two periods to the letter, creating a value such as "K.." — a valid retrieval parameter for companies beginning with the letter "K." The "Alpha Jump" macro (which I'll describe later) is then triggered to retrieve that range of records in alphabetical order.

If the range lookup finds no "K" records, the program bumps the "K" to "L" (the next alpha range), and loops back to the JumpNext field, re-executing the program to see if any "L" record exists. And so on. Here's the JumpNext program:

```
< @Play("Sound", "100, 5");
  If JumpNext = ">>" Then
  JumpNext = @Chr(@Asc(Company) + 1) + "ZZ";

  If @Asc(JumpNext) > 90 Then
  { JumpNext = "";
  @Msg(@Text(30," ") + "End of alphabet!"); Chome };

  If @Left(@XLookupR(@Fn, JumpNext, "Company",
  "Company"), 1) = @Left(JumpNext, 1)
  Then {
  JumpNext = @Left(JumpNext, 1) + "..";
  @Macro("Alpha Jump") }
  Else { JumpNext = @Chr(@Asc(JumpNext) + 1) + "ZZ";
  Goto JumpNext }
```

2. JumpSelect (<>)is the second Jump field. It displays the A through Z selection list. You simply click on the alphabetic range you want to jump to, and if a record in that range exists, the macro retrieves the range, displaying the lowest alphabetical record first. An @Msg message informs you if no records in the selected range exist.

Here's the JumpSelect program:

```
< JumpSelect = @Userselect(
  "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z");
  @Play("Sound", "100, 5");
  If @Left(@XLookupR(@Fn, JumpSelect + "ZZ", "Company",
     "Company"), 1) = @Left(JumpSelect, 1) Then {
  JumpSelect = JumpSelect + ".."; @Macro("Alpha Jump") }

  Else { @Msg("No " + JumpSelect + "... records found");
  JumpSelect = "<>"; Chome }
```

3. JumpPrev (<<) is the third Jump field. It performs like the JumpNext program, but in reverse — that is, it finds the next *lower* alphabetical range of companies. If you're already on an "A" record (ACME Plumbing), or no record exists lower than the range you're on, an @Msg message tells you you're at the end of the alphabet.

Notice that this program doesn't decrease the ASCII decimal value of the current company's first letter, or tack a "ZZ" onto it. It doesn't have to, because if you're on the Jiffy Products record, an @XLookupR on "J" can only return the next character lower than "J." While JumpNext moves up the alphabet one character at a time until it finds the next higher range of records, JumpPrev makes a beeline to the next available lower range.

Here's JumpPrev's program:

```
< @Play("Sound", "100, 5");
  JumpPrev = @Chr(@Asc(Company));

  If @Asc(JumpPrev) < 65 Then
  { JumpPrev = "";
  @Msg(@Text(30," ") + "End of alphabet!");
  Chome }

  Else
  JumpPrev = @Left(@XLookupR(@Fn, JumpPrev, "Company",
                "Company"), 1);
  If @Error Then {
  @Msg(@Text(30," ") + "End of alphabet!");
  JumpPrev = ""; Chome }
  Else {
  JumpPrev = JumpPrev + ".."; @Macro("Alpha Jump") }
```

The Alpha Jump macro

One macro — let's call it *Alpha Jump* — serves all three Jump fields. To record it, retrieve any record in the database, and move to the Company field. Press Shift-F2 for the Macro menu, select Define Macro, and press Enter. Don't assign a key identifier. With the recorder running, perform the following steps:

1. Press F11 to copy the field's contents to the clipboard.
2. Press F7 to display the Retrieve Spec.
3. Move to the Company field, and press F12 to paste.
4. Press F8 for the Sort Spec, move to the same field, and type *1as*.
5. Press F10 to execute the sorted retrieval. The same record should appear.
6. Press Shift-F2 to save the macro. Type *Alpha Jump* on the Macro name line, set Show screen to *No*, and leave the End with menu line blank. Press F10, then press Enter to save the macro to your default macro file (probably QAMACRO.ASC).

The macro should look like this in Write. The only difference will be the commands that move the cursor to the field on which you're searching and sorting:

```
<begdef><nokey><name>"Alpha<sp>Jump"<vidoff><f11><f7><dn><f12>
<f8><dn>1as<f10><enddef>
```

If you ever redesign the form and change the search field's position, you'll have to edit the macro to account for it. If you want the macro to show the records in Table View (to find the one you're after faster), see the sidebar, "Do It In Table View."

Test your alpha jump

When everything is ready, retrieve any record, and click on the ">>" (JumpNext) field. Depending on the number of records in the database, your PC's speed, and whether on a network, the first record in the next higher range of records should appear fairly quickly. (See the sidebar, "Boost Sorted Retrieval Speed.")

Next, click on the "<<" (JumpPrev) field, and you should be returned to the first alphabetical record in the range you started with. Click on the "<>" (JumpSelect) field, click on any letter, and the first alphabetical record in that range should appear. Regardless of the range, you can press F10 to move through the records.

Tom Marcellus is editor of *The Quick Answer*.

Boost Sorted Retrieval Speed

I keep a 10,000-record (18-field) address database to test techniques like this. Initially, my Pentium PC (with Q&A running in Windows 95) took an average of three seconds to jump from one alpha group of last names to another. (Keep in mind that the *Alpha Jump* macro is sorting the records before retrieving them.)

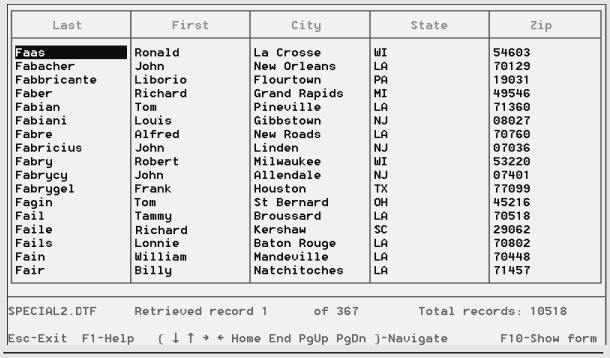
To see if I could reduce the lag, I performed a couple of tests. First, I ran a "null" Mass Update, using a blank Retrieve Spec, a Sort Spec of 1AS in the Last Name field, and a blank Update Spec. This reduced the lag to an average of two-seconds—a one-third improvement.

Next, I copied the database's design, then copied all 10,000 records, using 1AS in the Last Name field at the Sort Spec. With the new database's records now physically in alphabetical last name order, there was no alpha jump lag at all, and I could click through all 26 alphabetical ranges (A, B, C, through to Z) in less than 15 seconds.

Your results might not be as dramatic, but this shows how a little "special tuning" can improve sorted retrievals.

Do It In Table View

You can extend the *Alpha Jump* macro to show your records in Table View. This way, you can view 17 records per screen to more quickly find the one you need. (See Figure 2.) To do this, create and save a Table View Spec that puts the significant fields in the best column order for viewing (such as Last Name, followed by First Name, City, State, then Zip). Then, have the *Alpha Jump* macro, as its final actions, switch to Table View and select and run the saved Spec. The lowest alphabetical record in the range will appear at the top of the table, and you can press Page Down to move from one screen to the next. When you reach the record your after, press F10 to display it.



Last	First	City	State	Zip
Faas	Ronald	La Crosse	WI	54603
Fabacher	John	New Orleans	LA	70129
Fabbricante	Liborio	Flourtown	PA	19031
Faber	Richard	Grand Rapids	MI	49546
Fabian	Tom	Pineville	LA	71360
Fabiani	Louis	Gibbstown	NJ	08027
Fabre	Alfred	New Roads	LA	70760
Fabricius	John	Linden	NJ	07036
Fabry	Robert	Milwaukee	WI	53220
Fabrycy	John	Allendale	NJ	07401
Fabrygel	Frank	Houston	TX	77099
Fagin	Tom	St Bernard	OH	45216
Fail	Tammy	Broussard	LA	70518
Faile	Richard	Kershaw	SC	29062
Fails	Lonnie	Baton Rouge	LA	70802
Fain	William	Mandeville	LA	70448
Fair	Billy	Natchitoches	LA	71457

SPECIAL2.DTF Retrieved record 1 of 367 Total records: 10518

Esc-Exit F1-Help { ↓ ↑ → ← Home End PgUp PgDn }-Navigate F10-Show form

Figure 2. You can have your Alpha Jump macro switch to Table View after retrieving each group of sorted records. This way, you can view 17 records per screen.

The database featured in this article is available ready-to-use from Marble Publications for $24 postpaid. (Order disk No. **TM0897**.) To order by chargecard, call 800-780-5474 or fax to 301-424-1658. Or mail check to Marble at address on page 2.

Quick Tip

More on Variable Report Headers

In his May 1997 editorial, Tom Marcellus describes how to create a report with a variable header. The trick is a special header program that references a derived column. While trying it out, I discovered a sexy quirk. If the report includes a Page Break sort, the header will change on each page!

Suppose you have an employee attendance report. You concatenate the employee's last and first name in a derived column, sort on that column, and specify a page break so each employee's report prints on a new page. If you reference that derived column in the report's header, Q&A will print the employee's name in each page header. It's a terrific way to print a set of similar reports for distribution to different people. And it can help solve the *"Report too wide"* problem because you can make the derived column invisible.

Erika Yoxall, Hammer Data Systems, 330-527-4018

[For yet another way to take advantage of Q&A 5.0's variable report header capability, see "Special Date Calculation in Report Header" on page 11 of the July 1997 issue. —Ed.]

The Quick Answer

The Program Spec

Trapping Your Basic Data Entry Boo-Boos

JEFF NITKA

MY biggest client's sales force uses Q&A for order writing. Because the accuracy of the orders is vital, nearly every field in the order information database includes "traps" to prevent common data entry errors. I'll show you a few of these traps, and how they work. You can incorporate them in your own databases.

Can a numeric field store a fraction of a value?

If your customer orders items that must allow for fractions (such as when ordering 1.5 lbs.), format the pertinent quantity field (N) for numeric values. On the other hand, if the order quantity can't be a fraction (as when ordering units of an item), format the field (N0) for numeric values with no decimal places.

Can the quantity be less than or equal to zero?

Whatever your answer to this question, you might be tempted to handle it in the Restrict Values Spec. Doing so, however, only displays a warning if the value is out of range. (The user is still free to enter the value). To tighten this up a bit, use a program like one of the following to prevent users from entering a value less than 1:

```
Qty: > If Qty < 1 then Goto Qty else Cnext
```

```
Qty: > If Qty = "" then Goto Qty
   else if Qty < 0  then Qty = @Abs(Qty) else Cnext
```

Ensuring a Required field is always unique

If a Required field must contain a unique value, you could go to the Speed-Up Spec and type SU in it. Like the Restrict Spec, though, this method only displays a warning (in this case, if the value already exists in the same field in another record). The user can still override the warning and enter the duplicate value.

To guarantee that a value can't be entered more than once, you can use a program in addition to the Speed-Up Spec. Here's an example of a program for a unique account number field in a customer database:

```
Acct No:
> If Acct No = "" then Goto Acct No else
   If  @Xlu(@Fn,Acct No,"Acct No","Acct No") <> ""
   then { @Msgbox("This account number is being used.",
         "Please use a different account number.","");
         Acct No = ""; Goto Acct No }
  else Cnext
```

This program will produce the desired result when the user is adding a new record to the database. But it contains a serious flaw if the user is updating a record—that is, the @Msgbox command will always execute because the lookup will find the record on disk that represents the displayed one. To work around this, you can use a numeric, speedy, and read-only field named RecNo. When a new record is added, it's value is automatically incremented by one, like this:

```
RecNo: < If @Add then RecNo = @Number; Cnext
```

Now, you can revise the earlier program as follows:

```
Acct No:
> If Acct No = "" then Goto Acct No else
   If  @Xlu(@Fn,Acct No,"Acct No","Acct No") <> "" and
       @Xlu(@Fn,Acct No,"Acct No","RecNo") <> RecNo
  then { @MsgBox("This account number is being used.",
         "Please use a different account number.","");
         Acct No = ""; Goto Acct No }
  else Cnext
```

Protecting against typos

You can use Q&A's typecasting functions to verify that any data entered conforms to the type of field. For example, suppose a user types *Jan. 1, 1997* in a date field. (The period following *Jan* makes it unformattable as a valid date). Q&A displays a warning, but the user presses Enter to continue anyway. Try this:

```
Order Date: > If  @ToDate( Order Date ) = ""
            then Goto Order Date
```

To trap text entered in a quantity field, try this:

```
Qty: > If  @ToNumber( Qty ) = 0
     then Goto Qty
```

Trapping with an on-record-exit program

To add even more error-proofing, you can use an on-record-exit program to perform a final check on *every* field. (Have it duplicate the error-checking in the individual fields.) The reason is that the user could type erroneous data in a field, then save the record without first pressing Enter to activate the field's program. Without the final check, the erroneous data would be saved in the record.

See my Program Spec article in the July 1996 *Quick Answer* for a deeper discussion of on-record-exit programming.

Jeff Nitka develops Q&A applications part-time for Epoch Software, 908-874-3989. He's the author of the Program Evaluator (a Q&A program debugging utility), SurfDrve (a Q&A disk drive reader), and FaxMan (a Q&A faxing database), all available from Marble Publications.

The Quick Answer

A Database of Macros

PETER VENUTO

Why store macro names and descriptions in a database? There are advantages, including the ability to control which macros are accessible.

DURING last October's Q&A Masters Seminar and Users Group Bash, Bill Halpern and Gordon Meigs reported that macros run from Q&A's Alt-F2 macro list can be unreliable and cause the mysterious *500N* to appear on Q&A's Main menu.

For those who aren't aware of it, the appearance of the 500N can damage a database, cause the keyboard to lock up, and make Q&A "think" that a valid Program Spec isn't. Whenever that 500N appears, you should exit then restart Q&A.

To avoid using the Alt-F2 macro list, I drew on a technique Bill demonstrated—using a database as a replacement for unruly custom menus. In my case, I created a database of macros that I can lookup and run from any working database. Here's how to do it.

Design a database named MACROS.DTF with two fields, **Description** and **Macro Name**. Make **Description** a Speedy field containing a brief macro description (19 or fewer characters). You can prefix your descriptions with a letter or number, such as *A-Run Daily Report* or *01-Run Daily Report* if you want them to appear in a certain order on the macro selection list. In any case, you should include a macro description like *A-Cancel macro* or *01-Cancel macro* so it appears at the top of the list.

Make **Macro Name** a text field for the actual name assigned to the macro. Leave this field blank in the *A-Cancel macro* or *01-Cancel macro* record.

Add a record to MACROS.DTF for each macro in your macro file that you might want to run from a database.

Next, in a working database (a database from which you want to run macros), add a two-character field named **Macro**. You can place it near the bottom of the form to keep it out of the way.

Go to the Program Spec for this working database, and type this program in the **Macro** field:

```
< If @Askuser("Do you want to run a macro?","","")
  Then
  @Macro(@XuserselectR
  ("Macros", "Description", "Macro Name", "","",""))
  Else Cnext
```

When you move to the **Macro** field, the program confirms that you want to run a macro. If you answer yes, the macro descriptions in MACROS.DTF appear on a selection list. You choose one, and the @XuserselectR command returns its Macro Name to the @Macro command, which then runs the macro.

Options

The @XuserselectR command includes starting and ending range parameters (both are blank in the above example). You can take advantage of them by attaching prefixes to your macro descriptions in MACROS.DTF. This way, you can customize an @XuserselectR command for each database, so it displays only the macros you permit to be run from that database.

MACROS.DTF can itself be a "working" database. (That is, you can run your more sophisticated macros right from MACROS.DTF.) Simply set its Sharing Mode to *Allow*, and include the above program in it. For even more convenience, you can place a macro on the Q&A Main menu that opens MACROS.DTF and displays the macro list.

Careful attention must be paid to any macros that start from a database record and perform actions after leaving it. Be sure such macros know what to do with the record, such as saving it, or escaping from it without saving changes.

Peter A. Venuto is a veteran Q&A developer with expertise in accounting-related applications. peterv@mainsite.com.

The Quick Answer

Timely Payroll Solutions

DAVID FLAKS

Trouble with time-related calculations and retrievals? Try these clever techniques

DOS WIN

RECENTLY, I designed a payroll calculation database for a client. (see Figure 1.) Here are some of the problems I solved along the way.

Problem 1

The client always entered time-clock card data during the week following the payroll period, but not on any particular day of the week. So, I needed a way to make all the week ending days occur on a Sunday. I set the **Week Ending** field (#1) to @Date at the Set Initial Values Spec, then programmed it this way:

```
Week Ending:
< #1: If @Add and @Dow$(#1) <> "Sunday" then
  {#1 = #1 - 1; Goto #1}
```

This *loop* continues to deduct one day from the Week Ending date (@Date is the initial value) until it reaches a Sunday. This way, it doesn't matter which day of the following week the payroll data is entered—the Week Ending date is always set to the previous Sunday.

Problem 2

I set the morning and afternoon starting and ending times to default values at the Initial Value Spec. This saved the clerk from having to enter them unless they differed. I formatted these fields for time values so the Daily Total fields could be filled by simple calculation statements like this one:

```
#80 = (Morning Start - Morning End) +
      Evening Start - Evening End)
```

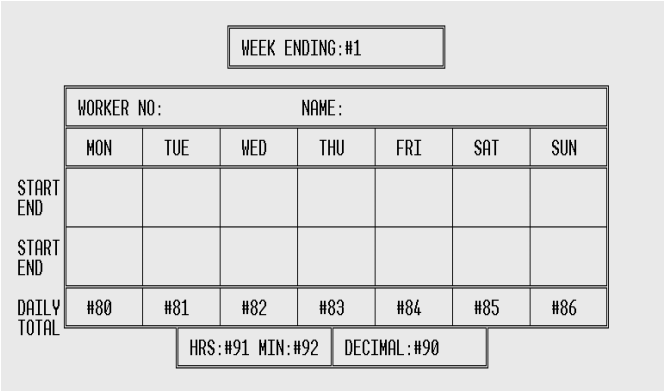


Figure 1. The payroll calculation database shown with the pertinent Program Spec field ID numbers.

A problem cropped up, though, when adding the total hours for the entire week. First, I tried using a time-formatted field to store the sum of the values on the Daily Total line. But when the total hours and minutes for the week exceeded 24:59, the program would fail. For example, if the weekly total amounted to 40 hours and 25 minutes, the result would be 16:25 (40:25 minus 24 hours) because there's no such time as 40:25.

The solution involved adding another field (#90) named **Decimal**, and programming it like this:

```
#90=(@Left(#80,(@Instr(#80,":")-1))+(@Right(#80,2)/60) +
     @Left(#81,(@Instr(#81,":")-1))+(@Right(#81,2)/60)+
     @Left(#82,(@Instr(#82,":")-1))+(@Right(#82,2)/60)+
     @Left(#83,(@Instr(#83,":")-1))+(@Right(#83,2)/60)+
     @Left(#84,(@Instr(#84,":")-1))+(@Right(#84,2)/60)+
     @Left(#85,(@Instr(#85,":")-1))+(@Right(#85,2)/60)+
     @Left(#86,(@Instr(#86,":")-1))+(@Right(#86,2)/60))
```

The program extracts from each field the whole number of hours. Because 8:30 and 16:30 are different length values, I used @Instr to find the position of the colon (":") so I could extract the digits to the left of it. I then added the two rightmost digits, then divided by 60 (minutes) to get decimal hours. The result yielded a decimal value of total hours for the week.

To extract the number of hours and minutes, I realized that the number of whole hours would be the integer portion of **Decimal**'s value, and the minutes would be its fraction multiplied by 60 minutes—thus these two statements:

```
Hrs: #91 = @INT(#90)
Min: #92 = (#90 - #91) * 60
```

By formatting the **Hrs** and **Min** fields as numbers with 0 decimal places (N0), I got perfect results.

Problem 3

As I mentioned earlier, each week's payroll was done on different days in the week following the payroll period. To print the checks, stubs, and reports, the clerk had to refer to a calender to find the previous Sunday's date (which, of course, differed each week), then enter that date into the Retrieve Spec's Week Ending field.

This wouldn't do. We needed a single Retrieve Spec that would *always* select the previous week's records. The following retrieval expression enabled the clerk to run

Continues on page 12

Macros—The Missing Link, Part 2

ERIKA YOXALL

Last month, you recorded and ran a simple macro. Now, find out how to use Q&A's Macro menu, and how to edit a macro in Write.

LAST month in Part 1, we created a simple database, then recorded a macro to open it in Add Data mode. This month, we'll use that database and macro to go into more detail on the Macro menu. I'll also show you how to read and edit a macro in Q&A's word processor.

Start by loading your test macro file:

1. Go to the Q&A Main menu, and press Shift-F2 to display the Macro menu.
2. Select Get Macros.
3. Type the path to your test macro file, such as C:\QA\MACTEST\QAMACRO.ASC, and press Enter.

Let's review the selections on the Macro menu. Press Shift-F2 again to display it. You'll see these choices:

Run Macro displays the list of available macros from the macro file currently in memory. (You can also display this list by pressing Alt-F2 or clicking your right mouse button, but due to reports that macros run this way can be unpredictable, I don't recommend it.)

Define Macro records a macro as you did with last month's test macro.

Delete Macro displays the list of available macros so you can select the one to permanently delete.

Get Macros lets you load a macro file.

Save Macros lets you save any macros you recorded during the current session but didn't save. Unless you create a macro for use *only* during the current session, it's a good idea to save it after recording it. Otherwise, you'll lose it once you exit Q&A. *If you save a macro to a file other than the current one, the file you save it to will be overwritten and contain only the newly-saved macro.*

Clear Macros removes all macros from memory. This doesn't delete the macro file, just removes it from memory.

Create Menu—Choose this option to create a custom menu. Q&A's custom menus can be quirky. Where possible, I recommend using the alternative "custom menu" technique described by Bill Halpern in the June 1996 *Quick Answer*.

When the Macro menu isn't enough

Although the best way to create a macro is usually by recording it via the Macro menu, you might need to edit it afterwards. For example, you can't record a macro that exits Q&A because as soon as you exit you lose the unsaved macro. Let's open our test macro file in Write to view the simple macro we created last month:

1. Go to the Q&A Main menu. Select Write, then Get.
2. Type the path to your test macro file, such as C:\QA\MACTEST\QAMACRO.ASC, and press Enter.
3. Select ASCII if it isn't your default document import type. The file should look like this:

```
<begdef><altl><name>"Leave<sp>a<sp>Message"<vidon>FA
<capsf4>C:\QA\MACTEST\ANSMACH<enter><enddef>
```

All macros contain the same elements:

<begdef> (*beg*in *def*inition) denotes the start of a macro.

<altl> is the macro's hotkey identifier. If the macro has no hotkey, you'll see **<nokey>** instead.

<name> indicates that the next sequence of characters is the macro's name.

"Leave<sp>a<sp>Message" is the macro name in quotation marks. This is the name you typed in the Macro Name field when you finished recording the macro. Note that a space is represented as **<sp>**. Macros don't contain any spaces.

<vidon> shows the changing screens as the macro runs. After recording your macro, select *Yes* at the Show Screen prompt for <vidon>, or select No for <vidoff>. You can always replace any vidon with vidoff to make a running macro look more professional, less "frantic."

FA<capsf4>C:\QA\MACTEST\ANSMACH<enter> are the keys the macro "presses." *FA* stands for File / Add data, and <*capsf4*> is the Shift-F4 key combination that clears the filename prompt. (Having a macro clear the filename prompt then type the path and filename helps ensure the macro opens the right file.)

<enddef> (*end def*inition) denotes the end of a macro.

If the macro ends by calling a custom menu, or contains a <wait> command (a pause in macro playback that allows you to enter text or make a selection), those elements will also appear in the macro. See the *Q&A Application Programming Tools Manual* for more detail.

Editing a macro

Suppose you want your *Leave a Message* macro to type your name into the new ANSMACH.DTF record. You could rerecord the macro using the Macro menu, but it's easier to edit it in Write so that it automatically types your name, then moves to the Message field. With the macro displayed in Write, follow these steps to edit the keys the macro "presses":

1. Place your cursor on the left angle bracket of <enddef>.
2. Make sure you're in Insert mode so you don't overwrite anything.
3. Type your first name, type <*sp*>, type your last name, then type <*enter*>. Don't press Enter at the end of a line. Let Q&A wrap it. Your macro should now look like this:

```
<begdef><altl><name>"Leave<sp>a<sp>Message"<vidon>FA
<capsf4>C:\QA\MACTEST\ANSMACH<enter>Erika<sp>Yoxall<enter><enddef>
```

4. Resave the macro file to ASCII file by pressing Ctrl-F8.
5. Now try the macro by continuing with these steps:
6. Return to the Q&A Main menu, and reload your test macro file by following the steps at the beginning of this article.
7. Press Alt-L to run the macro. It should open a new ANSMACH.DTF record, fill in your name, and move to the message field.

You can create a macro entirely in Write provided it contains all the elements in the right sequence and with the correct syntax. Usually, it's easier to record the macro, then use Write to make any changes to it.

For example, I usually specify <vidon> when I save a new macro so I can watch the screens change as it runs. When I'm satisfied it's working properly, I change it to <vidoff> in Write.

You can also use Write to troubleshoot a macro. By printing it out from Write (Ctrl-F2), and stepping through the macro's keystrokes one at a time, you can often spot where the problem is occurring.

In my final installment on macros, I'll show you how to create a more sophisticated macro that prints a report. I'll also show you how to make a macro pause while it's running, and prompt for user input.

Erika Yoxall owns Hammer Data Systems in Garrettsville, Ohio, specializing in Q&A and Microsoft Access. Phone/Fax 330-527-4018, emy103@worldnet.att.net.

Payroll . . . *continued from page 10*

the previous week's checks, stubs and reports without having to know the week ending date:

```
={@XLookupR("c:\qa\data\wages", @Date, "Week Ending",
  "Week Ending")}
```

The @XLookupR range command searches WAGES.DTF (the current database) for the records whose Week Ending date matches the current date (@DATE). If it doesn't find any (which it never does because the reports aren't run on Sunday), it selects the records with the next *lowest* date. And those records are always the right ones.

David Flaks owns DFC Computers in South Africa, specializing in custom Q&A applications, support, and training. dfc@pixie.co.za

---

MARBLE PUBLICATIONS

PO Box 9034
Gaithersburg, MD 20898-9034

Periodicals
Postage
PAID
at Rockville, MD