LETTERS

Letters to Visual Basic Programmer's Journal are welcome. To be considered for publication, letters must include your name, address, and daytime telephone number. Letters may be edited for form, fit, and style. Please send them to: Letters to the Editor, c/o Fawcette Technical Publications, 209 Hamilton Avenue, Palo Alto, CA 94301-2500. Fax them to 415-853-0230, or send them electronically to America Online at RobertVBPJ, or CompuServe at 74774,305.

SMART LUCK

The January 1995 issue of *VBPJ* was the first issue of your magazine I ever read. I bought it for the discussion on naming standards ["Consistent Naming Clarifies Code"] and the DDE primer ["Back to the DDE Basics"]. When I got reading it in more detail I found a gem, "Use DDE to Talk to Your Fax Program" [Getting Started with VBA].

Chris Barlow's article couldn't have come at a better time. I was just starting to do something similar the day I read the article. I had just installed Excel 5 at the office to learn VBA. My first VBA task was to fax out more than 100 35-page Excel 5 reports on a monthly basis.

I probably should have bought a lottery ticket the same day.

David J. Robertson
received on CompuServe

TRANSACTION TALES

The February issue of *VBPJ* had a nice article on conserving system resources ["Conserving Windows Resources," by Barry Seymour and Tim O'Pry, *VBPJ* February 1995], which seems to be an idea foreign to many VB developers. While this is information that needs to be shared every so often, I didn't really think I would find anything new. To my surprise, in a sidebar ["A Few Pointers"], I found that the authors were suggesting that using transactions in database code would eat up resources, and should be avoided. Knowing that transactions are an excellent way to speed up database operations, and having never had any resource problems (that I am aware of) related to them, I was skeptical, and I decided to check this out myself.

I created a small application that did two things: write 10,000 records to a single Access 2.0 table, and record system resource usage as it did so. The Access table consisted of a single Counter field and ten Text fields. I set the application up so that it would do its job both with and without transactions.

Basically, the procedure that added the records recorded the GDI, User, and free memory when it started. It then opened the database and table, issued an AddNew, added text to each of the text fields, and did an Update. Immediately before and after each Update, it recorded the GDI, User, and memory usage, and compared it against their initial values. The maximum drop for each value was then recorded. Finally, before the procedure ended, it recorded all three values again.

The results I received would seem to belie the article's suggestion that resources were wasted by transactions. Both GDI and User resource usage was immeasurable when the application was run with and without transactions. Memory usage was slightly higher at some point, but was returned at the end of the procedure. The results are tabulated below:

	Without Transactions	With Transactions
GDI before:	72%	72%
GDI after:	72%	72%
Max GDI used:	0%	0%
User before:	65%	65%
User after:	65%	65%
Max User used:	0%	0%
Max mem used:	523K	538K
Elapsed time:	9:22	2:48

As you can see, no more resources were used, but the processing time was cut dramatically. An added benefit is that an error handler can be added to discard any changes in the event something should go wrong by simply issuing a Rollback command.

I think readers would be better served by a comparison of the benefits and perils of using transactions, rather than categorically placing them on a "bad for your app" list.

Bob Dover
received on CompuServe

While transaction processing dramatically speeds up processing under the JET engine it does this by storing the individual transactions and then blasting them all to disk on the commit. How adversely this affects memory is totally dependent on the size and number of the records. Barry didn't say this was "bad for your app" or "avoid using it," he just

VISUAL BASIC PROGRAMMER'S JOURNAL

PUBLISHER & EDITOR
James E. Fawcette

ASSOCIATE PUBLISHER ADVERTISING DIRECTOR
Ellen Gould

ART AND PRODUCTION DIRECTOR
Michael Hollister

SENIOR DESIGNERS
Kristen Avery
Suzanne Whittaker

EXECUTIVE EDITORS
Matthew Carter
Frank Moncrief

MANAGING EDITOR
Christine Strehlo

ASSOCIATE MANAGING EDITOR
Nina Goldschlager

ASSOCIATE EDITOR
Robert W. Scoble

CONTRIBUTING EDITORS
Daniel Appleman , Chris Barlow
Andrew Brust, Carl Franklin, Roger Jennings
Craig Leach, Mark Novisoff, Sam Patterson
Keith Pleas, Richard Hale Shaw
Jonathan Wood, Jonathan Zuck

CONTRIBUTORS
Chris Adler, Deepak Agrawal
Steve Mann, Ken Schiff
Christine Solomon, Phil Weber

TECHNICAL REVIEW BOARD
Daniel Appleman, Andrew Brust
Ward Hitt, Deborah Kurata, Patrick O'Brien
Sam Patterson, Karl Peterson, William R. Vaughn
Phil Weber, Don Willits, Jonathan Wood, Jonathan Zuck

CIRCULATION MANAGER
Tena Larson

ASSOCIATE CIRCULATION MANAGER
Karen Koenen

CIRCULATION COORDINATOR
Shirley Modric

CONFERENCE DIRECTOR
Elaine Kato

CONFERENCE COORDINATORS
Julie Simms
Leslie Tighe

CATALOG SALES MANAGER
Craig R. Altman

ADVERTISING SALES REPRESENTATIVES
Cheri Poff
415-917-7650 or 800-848-5523
Carolyn Lighty, Andy Skurna
201-712-0044 or 800-685-2405

ADVERTISING SALES ASSOCIATE
Gwen Rambis

ANCILLARY PRODUCTS COORDINATOR
Andrew King

CUSTOMER SERVICE
Lara L. Ireland
Michael Sturdivant
Katie H. Sturdivant

ASSISTANT TO THE PUBLISHER
Trisha Merriam

RECEPTIONIST
Connie Pulmano

DESIGN TEAM
Enigma Inc.
"What We Do is Our Business."
(apologies to Andrew Layman)
Michael Carr, Cynthia Engler,
Linda Magyary, Kate Paddock,
Lynn Sanford

FAWCETTE TECHNICAL PUBLICATIONS

James. E. Fawcette
PRESIDENT

Wilson, Sonsini, Goodrich & Rosati
CORPORATE COUNSEL

HOME

advised people of the impact and to have this impact in mind when coding.

The point of this tip is for systems with minimal available memory. If you hard-code your transaction processing without taking memory into consideration, you can cause the system to crash. Out-of-memory errors can cause all kinds of strange and wondrous things to happen under Windows—and guess what? VB error trapping doesn't always trap these types of errors because the system crashes before the error code is invoked.

I always suggest that users make the number of records processed in the transaction dynamic, and compute the number based on the amount of free memory. This way, you get the maximum performance benefit for the individual system.—T.O.

ARE WE HAVING FUN YET?

I seldom do this, but I just wanted to comment on the February 1995 issue. I really enjoyed the article by Escher on Slimy Windows Hacks. It's nice to have a sort of offbeat article once in a while. It keeps the fun in programming and helps to keep us from taking ourselves too seriously. I guess that's why I sort of agree with the letter from Gary Alston [Letters, "Congrats and Critique," *VBPJ* February 1995] concerning diversity in articles. I would like to see at least one article per issue that was a little on the fun or different side. Keep up the good work.

Bill Booth
received on CompuServe

GOOD CALL

I wanted to write to thank you for publishing the article "Uncommon Dialogs" by Jonathan Zuck [Windows Programming, *VBPJ* February 1995]. I had been considering writing a set of VB functions to replace the COMMDLG.VBX (just another thing to worry about when distributing) and this article gave me the motivation to finally do so.

However, I was a bit disappointed to read that the GetOpenFileName API function requires an address to a variable and so requires a DLL such as *VBPJ*UTIL.DLL mentioned in the article. Although the *VBPJ*UTIL.DLL is very small, it's still another file I'd have to worry about when distributing my applications. I wondered if some of your readers might be in the same position as me so I thought I'd let you know another way to get the address of a variable.

It seems the VBRUN300.DLL contains a function that does just this. Here's the declaration:

```
Declare Function VarPtr Lib _
   "vbrun300.dll" (var As Any) As Long
```

Or, if you declare it with an alias like this, you can slip it right into Jonathan's code:

```
Declare Function SSegAdd Lib _
   "vbrun300.dll" Alias "VarPtr" _
   (var As Any) As Long
```

I suppose this function might be removed from the VBRUN400.DLL but when I convert I'll probably have other things to change as well. I wish I could take credit for being the clever one who found this function but I can't. It was suggested by Tom Metro via e-mail way back when we were talking about the VBRUN200.DLL. Perhaps there are more useful functions in the VBRUN300.DLL. Has anyone ever tried to document them?

Kyle Lutes
Little Rock, Arkansas

Wow! Thanks for writing in! I once spent a great deal of time poring over VBRUN100.DLL to see if there were any functions callable from VB and found none. I never checked back or I would have found this incredibly useful function. I did check again now and this is sadly the only call-

able function that is in there but this is a gem. Forget the DLL I wrote, this is the way to go. Thanks again and thanks to Tom Metro, wherever you are. You got me on this one but I couldn't be happier.—J.Z.

ERGONOMICALLY INCORRECT

I enjoyed the Windows 95 Style article ["Design Windows 95-Style GUIs Now," *VBPJ* March 1995]. It was written in a clear manner by someone obviously close to the design decisions.

What was missing from the article (and is also missing from EVERY discussion I have encountered about design guidelines for Windows 95 software) is recognition that the Windows interface responds to physical movements of our hands, arms, and fingers as we manipulate its controls with the pointing device (usually a mouse, but often a torsion-sensing "stick").

For instance, nowhere in the Microsoft design guidelines do the "ergonomics" of mouse movements influence the recommended placement of buttons on a dialog. We all hit Cancel when we mean OK because our physical tendency is to underestimate the distance we have to move the mouse to hit OK. When we're in a hurry, that shortfall in our physical movement frequently translates to the location of the !@#$ Cancel button (which was placed there in response to an aesthetic or visual recognition decision, not a physical design decision).

Consider the consequences if the ignition switch on your car was as near to the gas pedal as Microsoft recommends you place Cancel near to OK (I'd say six pixels equals about two inches in the car).

Visual design guidelines for software should reflect the angles and sweeps of our bodies as we manipulate the pointing devices to select controls. Designers of mechanical devices ranging from bicycles, motorcycles, and autos to sailboats and airplanes consider our physical movements in the design of controls, why shouldn't software engineers? (And yes, this means we need to design differently for left-handed users!)

I predict that the most aggravating thing most of us will find in Windows 95 is the placement of the Close window button (the "x" immediately to the right of the Maximize button) . . . Think for a moment: how often will you hit it by mistake when you meant to click Maximize? The three little buttons look really nice all in a row up there on the upper-right corner. But Close has such dramatically different consequences from the other two controls that it should be separated from them by a radius at least as great as the inaccuracy of an average user's mouse movements . . . as in Windows 3.1, for instance . . . lest we hit it by accident.

Thanks for listening!

John Cantrell
Winter Springs, Florida

LAUDING LINKED LISTS

Finally, real coding for real programmers ["Code for Speed," *VBPJ* March 1995]. I've been coding linked lists in VB ever since dynamic arrays were introduced in VB 2.0. I've even posted code in MSBASIC two years ago (or so) demonstrating the technique. I was pretty much run out of town and ignored. They were even demeaned as fancy array-handling techniques which couldn't be real linked lists because pointers weren't supported in VB. Basically (pardon the pun) I said 'hogwash' and posted them anyway.

I have even written routines to simulate a stack (in Visual Basic) since VB is limited as to how deep a stack can go without running out of stack space (I needed a way to handle binary trees in an array structure for building outlines).

One of the biggest problems most VB programmers face can be solved by linked lists. This is when a huge pick list is created and the developer has allowed for multiple selects to occur. Instead of searching the

entire list box to find the selected members why not insert each user selection into a linked list and remove each item from the list that the user deselects as each Click or MouseDown event occurs (be sure to check the Selected property of the item to know if it is to be added or removed). The ItemData array property can be used to map a selected item's entry in the linked list array.

In closing, a lot of developers seem to feel that it is not worth the effort it takes to create a linked list in VB (or other structures for that matter), especially when there are so many cool components available. Personally I think that using too many VBXs can be just as detrimental as not using enough or any. To VB developers everywhere just pick a core set of VBXs (one of which needs to include a tool to allow for hooking Windows messages—my favorite is SpyWorks-VB from Desaware) and stick with them. Otherwise you will find that not all VBXs work well together or are created equal.

Craig Burkett
Winston-Salem, North Carolina

HIP ON TIPS

Yesterday's mail brought the first copy of my new subscription to *VBPJ*, which I ordered sight unseen. After looking it over, and reading the Tips supplement ["99 Really Hot Tech Tips For Serious Programmers," *VBPJ* Supplement to the March issue, 1995] all I can say is, "Thank you thank you thank you thank you thank you thank you. Thank you thank you thank you."

If subsequent issues are as good as this one is, have no doubt that I will renew my subscription. I've been subscribing to computer publications since 1979; I have yet to see a general-market product so full of useful information as *VBPJ* is.

Robin Sheppard
Seaside, Oregon

Thanks for the bonus of the 99 tips. Effective documentation like you provided, without a lot of advertising, is especially helpful to me, particularly in the late hours, with two or three reference manuals on the floor, the coffee cold, the eyes bleary, the body tired, everybody in bed already asleep (including the dog!), but with the knowledge that there HAS to be a way to efficiently do what I need to do.

I hope you continue to try different formats and ideas to "spread the word." Thanks, also, to all those who took the time to contribute their ideas and experiences. Your professionalism as demonstrated by the sharing of these ideas is really appreciated.

Jim Orr
received on the VBPJ *Forum on CompuServe*

Your March 1995 Supplement to *VBPJ* was right on the money. Although I've been programming in VB for 1 1/2 years now and thought I knew it all, I found six or eight tips I could make immediate use of as well as some to note for future reference.

My personal favorites were "Clear Out Data When Unloading Forms" [p. 11], "Preventing 'Invalid Use of Null' Errors" [p. 16] (boy did this get rid of a few lines of IsNull checks!), "Avoid Using Variants" [p. 28], and "Indenting Code Blocks" [p. 22]. I'll have to disagree with the "Use the Len() Function to Check for Empty Strings," [p. 21] though. The small amount of run time saved vs. time spent figuring out what the heck you meant when you wrote the code, especially when you're maintaining code with a group of programmers and you need the extra readability, ain't worth it.

Also worth noting is that "Make a Read-Only Text Box Without Graying the Text" [p. 21] still permits the read-only text boxes to receive focus which may or may not be desirable. I achieved the same result by sending focus to an always enabled control in each of the GotFocus events for controls I wanted disabled but not grayed.

This took a few more lines of code, but it prevented users from clicking to the disabled controls. I had already figured out "Close VB Before Compiling" [p. 2] but it sure was nice to see in print. Try explaining, "Can't insert or paste - data too long for field" errors that magically disappear upon recompiling to end users. They think you won't own up to bugs in your code!

Thanks y'all for the tips. Your magazine is always worth the money!

Shirley McMahon
Paducah, Kentucky

We would like to correct an error in the "Indenting Code Blocks" tip. It should read: "You can highlight a block of text in the code window and press the Tab key to indent the block or Shift-Tab to unindent it."—Eds.

I'm sitting at my station when the boss comes up and says, "Make our Executive Information System call and maximize an iconed instance if the user tries to fire a second instance." He then flops a photocopy of page 10 from "99 Really Hot Tech Tips for Serious Programmers" on my desk and points to the tip "Checking for a Previous Instance"! That's something I had never done in VB and that tip was worth the subscription price to *VBPJ*! (When I got home the first issue of my subscription to *VBPJ*, plus the tips, was waiting on my desk!) As it turns out the tip didn't work right because the Unload Me instruction had to be replaced by an End instruction. Also, the tip only worked when the first instance was either iconized or at least the first Form_Load event had completed loading. In our app (not using Sub Main) a modal login form is displayed half way through the first Form_Load event. If a user doesn't respond to the password text box and leaves it "hanging" while focusing on another application, then forgetting about the first instance attempts to fire a second instance, the AppActivate function will cause an error #5, illegal function call. And, if you don't shut off your error handlers before you end the second instance you are occasionally greeted with a GPF.

So, that tip led to some profitable learning experiences. BTW, I've been programming computers since plugging jumpers on an IBM 402 Tabulator was considered programming. You are never to old to learn!

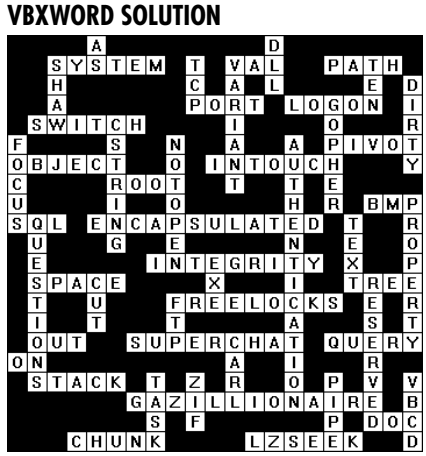
Jerry Kreps
received on CompuServe

I would like to see you try to organize tips by subject area. The 99 tips was great! I would also like to see something like a monthly dozen. "DDE Tips," "Database Tips," "Setup Tips," "xClipper Tips" <bg>, etc. with a different focus each month. I think it would make it easier to track down a tip you saw SOMEWHERE.

Just a suggestion, you guys are GREAT!

Kathleen Joeris
received on the VBPJ *Forum on CompuServe*

VBXWORD SOLUTION



Here is the solution to VBXword, the crossword puzzle that appeared in the April issue of *VBPJ*. Congratulations to Jeff Harding, John McCormick, and Nagarajan Krishnamurthy, who each won *VBPJ* prizes for their March puzzle entries.