Graphic Design by Dave Sherman

VisualBasic FAQ (Frequently Asked Questions) Revision 1b

Table of Contents

Table of Contents - Continued

Table of Contents - Continued

I. Preface

A. Who made this FAQ?

This FAQ is the result of a collaborative effort between four VB professionals. They are Dave Sherman, Jim Houghtaling, Charles Haeberle, and Guy Thompson. Below is some information on each of the authors.

David Sherman is a professional living near Cleveland, Ohio. He has been programming now for well over a decade, and programming for a living for over 5 years first as an Independent Computer Consultant and then as the Owner / Lead Developer at Excelsior Software Productions. During this time he has worked with over 10 different programming languages, but is proud to call VB his favorite. Dave has been an avid Visual Basic Programmer since VB 1.0 first came out. He is also one of the two founders of #VisualBasic, of which he is very proud.

Jim Houghtaling has been cursed with curiosity about PC's since 1983 and has never been the same. He spent 4 years working on PC hardware in the military and has been fixing and programming PC's ever since. He discovered Visual Basic and was captivated by the ease of use and flexibility. Currently he works for a small company as the lead developer for multi-user client/server applications and Hand-held computers.

Charles M. Haeberle is a Microsoft Certified Solution Developer who works for Software Consortium, Inc., a consulting firm in Towson, MD. He has done work for USF&G, T. Rowe Price, NationsBank and Blue Cross Blue Shield, to name a few. His last project, a teamwork effort, was nominated for the Microsoft Best Application award.

Guy Thompson is lead programmer at J. Goodman and Associates, Inc, a company specializing in document management and imaging software located in Memphis, TN. A professional programmer since 1984, he has watched the PC "grow up" (He claims to still have a copy of Windows 1.0 on 5 1/4" floppy discs). He has been programming in Visual Basic for a little over 3 years now, starting out way back in the dark ages of VB 1.0.

B. Where can I get a copy?

We are pleased to announce our partnerships with the following web sites, organizations, and companies all dedicated to helping us better distribute this FAQ to the Visual Basic Community. Please express your thanks to them for all of their help and hard work.

You can find the latest revision of the FAQ published on the following Quality Web Sites.

Association of Visual Basic Professionals
http://www.visualbasic.org/

Carl & Gary's Visual Basic Home Page
http://www.cgvb.com/vbfaq/

Que Publishing's Visual Basic 5 Night School
http://www.mcp.com/que/nightschool/faq/

Visual Basic Help
http://www.awod.com/gallery/rwav/robertjh/vbasic/vbfaq/vbfaq.htm

Visual Basic Broad-Band Network
http://www.geocities.com/SiliconValley/4548/vbfaq.htm

You can also find the FAQ published with/in the following professional publications.

Visual Basic 5 Night School
Que Publishing
ISBN: 0-7897-0921-X
http://www.mcp.com/que/books/descriptions/0921xd.html

II. General Questions

A. What is Visual Basic?

What is Visual Basic? Only the worlds most widely used development language. What does this mean? Well quite basically a person or a team of people wrote all applications you run on your computer. Visual Basic is one of many programming languages that people use to make those applications.

Microsoft announced Visual Basic 1.0 in May of 1991, since then windows application development has never been the same. Visual Basic was the first language to introduce Rapid Application Development (RAD) for the Windows platform. It integrated a very powerful engine with an environment that made the programmers life a lot easier. Allowing them to develop their windows applications 20-80% faster then any other method available.

During it’s short life Visual Basic has continued to break records and set forth many new standards to which all other languages would conform. As a matter of fact, Visual Basic during it’s almost 6 years in the market place has outsold ALL other programming languages, many of which had been around for 10+ years longer. Even today as new programming languages, and new versions of older ones, are introduced most of them are based upon the original concepts Microsoft developed and Visual Basic introduced to the world. A couple good examples being Delphi and VC++, both are visual development environments that owe very much of their existence to Microsoft when it set the standards and broke all rules with VB 1.0.

Many die-hard programmers that use other languages will testify to the death they owe nothing to VB and VB isn’t a very good language, but they are just fooling themselves. Microsoft since day one has had big plans for Visual Basic it has introduced countless new standards and countless new ideas to the programming community. Today we find Microsoft integrating Visual Basic into all of their MSOffice programs. We see VB taking on wings as it migrates to the World Wide Web with VBScript. We find Visual Basic still breaking the rules and continuing to amaze us.

One of the most disputed facts is how much of a following Visual Basic actually has in the world. Many VB programmers are constantly put down because, even today, some see it as an inferior or beginner language. In reality Visual Basic has come a long way since the early days and has greatly matured as a language to now be a fully functional and advanced development language ready to tackle anything you throw at it. There is no need for VB developers to hide or be ashamed of using VB just because the word “Basic” is in its name. In reality VB is anything but “basic” nowadays. Yes it’s still the easiest windows language to learn and use, but it also has the power and advanced features for the real programming pros that makes it an all around winner. Creating apps on a very rapid scale and doing so easily, creating very powerful apps that will integrate perfectly with windows, the Internet, and hundreds of other professional applications are just a couple reasons why developers continue to migrate to using Visual Basic.

Next time someone comes up to you saying VB isn’t a professional language, well try tossing back a couple of these indisputable facts. You must excuse me for the links proving these but many of the direct links to the statistics are outdated when I went looking for them. With the frequency MS and the independent research companies update their sites they seem to have gotten lost in the shuffle (then again it HAS been 6 months.) So all of these are links to professional business sites that still refer in part or in whole to the information.

> When Visual Basic 4.0 was released it broke ALL records set by all other languages for sales. Within less than one year of being on the market over 1 million copies had been sold.
>
> > IDG Books
> > Visual Basic Secrets
> > http://www.ishops.com/softpro/1-56884-872-2.html
> >
> > > Visual Basic 4 is now the best-selling development tool of all time, with over a million copies sold. There is no better way for users to master the vast system of extensions, techniques, and third-party tools that have developed along with this complex programming language than with this hot title. A bonus CD-ROM comes packed with all the book's working code samples, plus many third-party developer products and custom controls.

There are approx. 4.5 million developers in the world that make windows applications.

Silicon Investor
A Silicon Investor Pick for '96
http://techstocks.com/picks96/Jan_9.html

"Microsoft's role is to help ensure that the 150 million users of Windows and the 4.5 million developers for Windows have what they need to ride the Internet tidal wave."
- Bill Gates

There are more then 3 million developers using Visual Basic to create their windows based applications. Do the math 4.5 million – 3 million = 1.5 million not using VB. Granted you will have some developers that will overlap using VB equally with some other language such as Delphi or VC. But we are focusing on what language is used most by each developer. Most developers know more then one language but use one a lot more then the others. When you compare the VB developer numbers vs. the numbers one gets from the other languages there just is no comparison.

http://www.summsoft.inter.net/vba/vba5faq.htm
http://205.185.183.34/html/e1_060496a.html

"More than 3 million developers are actively creating applications with Visual Basic and Visual Basic for Applications, making this technology the most widely used development environment in the computing industry," said Ted Johnson, Visio co-founder and vice president of worldwide products. "Our early commitment to ActiveX technology has played a critical role in establishing Visio as a graphics solution platform. Now, by providing customers with the same Visual Basic language technology used in the Microsoft Office suite, we can share our Visio-based developer expertise more efficiently with a broader audience, and one that is already familiar with VBA as a development language."

In the end the recurring phrase you find is " Visual Basic is the worlds most widely used programming system. " I could only find one reference still on MS's site that hasn't been redone 50 million times that still had any part of the statistics. This is a very bad example it doesn't have anything in it really showing the numbers but it does none the less have the quote saying VB is the most used. Coming from MS that means a lot considering they make many of the other languages such as VC++.

http://www.microsoft.com/hk/press/prvbs.htm

There are literally thousands of other references throughout the web giving statistics and greater information on the subject. I wish I had saved some of the statistical htmls a few months back so I could show you the raw numbers, but I didn't bother thinking I would ever need to list them anywhere. If you wish more information on VB its usage statistics etc do some web searches or call Microsoft, I'm quite sure they would be more then happy to give you the statistics.

- Dave Sherman

B. Where can I get Visual Basic?

To order Microsoft Visual Basic or other Microsoft Visual Tools, or to receive a reseller referral, in the United States or Canada, call 1-800-621-7930, Dept. A334WB. Outside the United States and Canada, please contact your local Microsoft subsidiary.

C. What is the latest version, and is it worth upgrading?

The latest version of Visual Basic is currently 5.0. Visual Basic 5.0 offers many new functions and options too numerous to mention all of them here. When you look at all of the changes, two of them stand out as being the biggest improvements.

Visual Basic 5.0 now has a native compile mode. What this means is VB will create fully compiled exes rather then p-coded ones. Visual Basic does this by exporting your project to an intermediate language that is then compiled using a c++ compiler. So in the end what we find ourselves with is a very fast performing exe. Don't let this fool you though, the p-code compile mode is also much faster then it was in VB4 and the previous versions. From a standpoint of which mode to compile your programs in though, the native compile mode yields program performance results about 20% faster then the p-code compile in vb5 but both are extremely fast.

Probably the second biggest improvement is the ability to now create ActiveX controls and documents. For quite some time now programmers have been limited into being forced to use VC++ or some other language to do this. With VB5 this is an issue no more. Create ActiveX controls quickly using the development tools provided with Visual Basic 5.0. Creating controls in VB5 yields pretty much the same results as you get by creating them in VC++ except you can develop them many times faster in VB5. Remember the ActiveX controls are compiled using the native compile mode, using the same compiler VC++ uses to compile its programs and ActiveX components.

In the end in upgrading to VB5 you will notice huge performance gains in just about every area. VB5's functions and methods benchmark normally much faster then those of VB4's or VB3's. When you add the speed enhancements/performance gains you see in your projects, with being able to develop your apps faster then ever before, with being able to use/access more functions and more tools then ever before there just really is no comparison. If you are still unsure if upgrading is the right solution for you, you might want to take a trip on over to MS's site, below you will find some links to find more information on VB5.

What's New in Visual Basic 5.0?
http://www.microsoft.com/vbasic/docs/vb5new/vb5new.htm

Top 20 New Features in VB 5.0
http://www.microsoft.com/vbasic/docs/vb5new/vb5new1.htm

What's New in Internet Development?
http://www.microsoft.com/vbasic/docs/vb5new/vb5new2.htm

What's New in the Development Environment?
http://www.microsoft.com/vbasic/docs/vb5new/vb5new3.htm

What's New in Data Access?
http://www.microsoft.com/vbasic/docs/vb5new/vb5new6.htm

You can find this information and much more on MS's site. If you are unsure of upgrading or just curious about the new features take a trip on over, it's just a click away.

D. Why VB? How does it compare to other languages?

First off this section is NOT intended to start any programming language wars. It is intended to provide the users with statistical facts and present them with information from some of the leading professionals in the business, allowing the reader to make their own choices.

What do you think is the most important factor that effects a program's performance? Just a hint, if you think it's the language its made in you are wrong. While the language does play an integral role in a programs performance it is more the programmer's experience that a program if effected by. A programmer of many years has the experience needed to optimize and create his programs better then a programmer of a few years. I have seen expert VB 4.0 programmers create programs that rival and surpass many Delphi 2.0 and even some VC++4.0 programs. VB 4.0 in many benchmarks shows itself to be slower then these other languages, so how is it possible a program made in it can be faster? Plain and simple, the programmers who made those VB programs had many years under their belts and knew exactly how to optimize their programs and use little tricks/trade secrets to speed up their applications tremendously. An expert programmer can make any language a winner really, so the argument over which programming language performs better is really not a good argument. Because it's the programmer or team of programmers that make or break the program, not the language the program was written in.

March 17, 1997
Looking Forward
'Smaller in Java' doesn't always mean better
By Mark L. Van Name and Bill Catchings
http://www.pcweek.com/opinion/0317/17cvn.html

> ....
> Just because an application was developed in Java [or any other language], however, does not mean that it is a better program than one written in another language. Program quality depends less on the language than on the skill of the developers, and no language is a guarantee of good code.
>
> Similarly, the smaller of two applications is by no means necessarily the better. The larger program might simply provide more useful features, or provide the same features in a better way. Being larger, of course, is also no assurance of being better—you cannot judge application quality by size alone.
>
> Which brings us to the last linchpin of this pitch: Fewer features are better, because most people never use most features. What's wrong with this argument is that the market has voted once on this topic, and it chose the large, feature-rich applications.
> ....

This being said for those that want more information on the actual language engine speeds continue reading, for those that don't you can just skip this part.

The competition in the programming language industry can get to be quite annoying with the same arguments constantly being brought up. Personally I've gotten quite disgusted with all of the misinformation, lies, slander, and propaganda passed about. Most of it seems wagered against Microsoft and Visual Basic by people who I can only assume do not bother to educate themselves to the facts for one reason or another.

VB vs. Delphi

On paper the major competitor of Visual Basic 5.0 is Delphi 2.0, I say on paper because it has been written up as such in many places but in reality Delphi never has posed much of a threat to Visual Basic mostly because of market place. Delphi programmers and programs have very little market, the average amount Delphi programmers are paid isn't very much, but most of all the big corporations didn't want to invest millions of dollars training their huge programming departments in a language that never had much backing to begin with.

Looking closely we find Delphi really isn't that bad of a language. In fact Delphi 2.0's speed was very close to that of VB4.0's, each had it's own strong points and weak points and they balanced out for the most part. Since then however, Visual Basic 5.0 has hit the scene with a big bang.

Carnegie Technologies, a premier provider of leading edge information technology services, preformed a study comparing VB4, VB5, Delphi2, And PowerBuilder5. Out of the 12 tests they preformed, VB5 ranked in first place in 8/12, faster then Delphi 2.0 in 9/12, and faster then VB4 in 11/12. Out of the 4 tests VB5 didn't take first place in, it ranked second place in half of those four, and in the test where it ranked slower then VB4 it was slower by under .040 of a millisecond.

	Test	Visual Basic 5.0	Delphi 2.0
Display	Empty Display	52.509	26.851
	Labels	.316	1.295
	Bitmaps	1.226	3.296
Database	Overhead	9.745	7.838
	Small Retrieval	.340	.612
	Large Retrieval	.251	.110
OLE Automation	In Process Let	.000294	.084
	In Process Method	.000295	.082
	Out of Process Let	1.066	2.572
	Out of Process Method	1.039	2.572
Language	Sieve of Eratosthenes	360.625	3323.82
	String Manipulation	900.35	48.13

http://www.microsoft.com/vbasic/docs/vb5bench.htm

In overall performance, as shown by the Sieve of Eratosthenes test, they found VB 5.0 to be approx. 10 times faster then Delphi 2.0, and over 15 times faster then VB 4.0.

At the end of their report, Carnegie Technologies reminds the reader of the many other points developers should consider when choosing a language.

You can view the report for yourself at Microsoft's web site, just follow the link under the table above.

Despite Delphi's actual performance the biggest blow to Delphi's standing is oddly enough, Borland who makes it. Borland is in some very serious financial problems right now and many analysts are expecting them not to be able to pull out of it. They lost 3 top executives last year, including one who left to join Microsoft, and their problems continue to worsen today. The last quarter they had a net loss of almost 10 million, and made about 15 million less then the previous year. Because of all of the loses they were forced to cut over 15% of the work force at their headquarters. A good part of the problem according to Borland is their existing products, such as Delphi, aren't selling in the market with the fierce competition it's getting from Microsoft's Visual Basic and Powersoft's Powerbuilder.

Losses, layoffs at Borland
http://www.news.com/News/Item/0,4,4540,00.html
Borland posts $9.8 million loss
http://www.news.com/News/Item/0,4,4797,00.html
Borland loses exec to Microsoft
http://www.news.com/News/Item/0,4,3814,00.html

The above articles show Borland is in a lot of turmoil. What does this mean to the Delphi programmers and the language itself? Nothing good that's for sure. The stability of the company that makes a language and how much corporate backing a language has greatly effects it success and sales. Look at it this way, would you spend millions training your employees in a product from a company that won't be there next year?

There are two possible outcomes for Borland here. If they go out of business the language will undoubtedly die as well. Programming languages are constantly changing to meet the abilities the new computers and operating systems provide. If Borland goes out of business who is going to make the new versions? No matter how good a language is, it can't survive long without the company to make new versions, provide technical support, and fix bugs.

On the flip side, if Borland does manage to delay or even turn-around the current trend they are on, they have already lost millions and millions of dollars and cut 15% of their work staff from their headquarters. With less programmers and less money to use towards development of new / existing products what is going to happen? Can a language continue to be revised as often without the programmers to do it? Can a language be upgraded and new things added to it without the work staff to research and develop these new technologies? Lastly without as much money to devote to their projects will they be of the same quality?

VB vs. JAVA

There really isn't too much to say here. In the VBScript vs. JAVA Script category only time will tell which is accepted to be the better standard.

We can look at it from a different perspective however, compared to a fully compiled ActiveX Add-In and applications that integrate the Internet and the VBA engine into them, an interpretive language is not the best available solution.

Silicon Investor
A Silicon Investor Pick for '96
http://techstocks.com/picks96/Jan_9.html

> "Yes, they licensed Java... they had to in order to prevent a riot on Wall St. The real story is: (1) Java is just a programming language (2) Java is pathetically slow - interpreted! (3) Visual Basic will do all that Java can do and more (4) VB is integrated into Excel, Word, etc. Imagine being able to buy an Excel worksheet that has built-in Internet hookups to load information off the net (e.g. stock prices, TV shows, etc.) Imagine a Word document that could do the same thing! The whole idea of a browser is archaic... each application should be Internet-ready. Coming soon from MSFT!" - Steve Lerner

VB vs. VC++

In reality VB 5.0's major competitor is VC++ 5.0. Many will say, " You can't compare those, it's like comparing apples and oranges." They'd be right, the two languages are based on totally different structures and concepts. So what we will look at instead is the future of each, their role in the programming/world community, their ease of use, how rapidly you can develop your robust programs in them, and the performance one will get out of each. Visual C++ is made by Microsoft just like Visual Basic so both languages have a very stable company making them and providing support for them so that isn't really an issue discussed in this comparison.

Microsoft is constantly taking on new frontiers and is continually expanding its market. Much of the future of the World's computers relies quite heavily on Microsoft. We all go where Microsoft takes us. Microsoft's catch-phrase, "Where do you want to go today?" takes on new meaning when you consider how much of an effect MS has on the world, it's computers, and the future of both. So it makes a great amount of sense that which ever products MS places in the forefront of their plans, as they relate to their other products and the companies / worlds future software, those products would take off in more then one way.

Visual Basic is the language that has stepped up to the plate on this one. Microsoft chose it as the language to integrate with all their MSOffice programs. But not only is MS backing and supporting VBA5 as the language of choice to integrate with it's software but so are 50 of the MAJOR commercial software development firms in the world. Leading companies such as Adobe, Autodesk, Texas Instruments, Rockwell, Seagate Software, NetManage, McAfee, Symantec Corp, Micrografx, Visio, and many more have all licensed it for use in their applications. Visual Basic programmers can now not only write stand alone solutions, but now they can create integrated solutions within many other software packages. To see what some of these companies have to say about VBA5 visit the following link:

http://www.microsoft.com/vba/license/vbawho.htm

With the rapid growth of the Internet Visual Basic has also found a home developing for it, integrating with it, and redefining it as we know it. VB is taking on many areas previously thought to be only for VC++/C++. But unlike those languages VB has gone beyond that to be integrated into the web and many other places. VBScript jumped up to the challenge and has been in use on the World Wide Web proving itself ever since. Areas such as creating ActiveX controls/documents for use in programs and on the WWW was formerly not something to even be considered in VB. With VB5 however, MS added full support so now VB programmers can create ActiveX Controls, Documents, and DLLs.

In the long-standing tradition of BASIC, VB continues to provide a very quick and easy to use programming environment and language. This makes it great for the beginners because it’s so easy for them to learn. However unlike the BASIC’s of the past, Visual Basic provides an exceptional amount of advanced functions, methods, and routines. These provide the expert with many options, add to that the ease of the language and the expert’s job is quite faster because most of the structures/coding require a very small amount of work to accomplish the same task that could take pages in other languages. All of this coupled make VB an excellent choice for both the beginner and expert alike.

VC++ on the other hand is more geared towards the expert, making it a very tough language to learn and not at all a language for anyone new to programming. VC++/C++ can be quite cryptic; this makes coding a rather slow process for the experts as well as for the beginners. VC++’s cryptic code does have a purpose to it; many feel it is a form of job security because in many instances only the original author is able to decrypt the meaning/purpose of the code unless it is commented extraordinarily well. VC++ also does provide a more direct way of being able to access a computers lower-level functions, but those same functions can still be accessed in other languages directly or via DLLs. VC++ is a very good choice for C++ programmers looking for the next logical step or looking to migrate to the windows platform.

Lastly we look at the performance of VC++ 5.0 exe’s vs. VB 5.0 exe’s. We find that the performance between the two compilations is almost exactly the same. In fact it’s quite comparable to the performance we saw back with VB4 and Delphi, each one has it’s own strengths where it is faster then the other but in the end they balance out. According to the some professional developers that use both, they have made it quite clear they have seen no performance difference except when accessing low-level functions. Actually there is a very good reason for the performance being so close in benchmarks. When you compile and exe in VB5 using the native compile mode it actually exports your project to an intermediate language that is read in, supposedly, by the same C++ compiler and linker VC++ uses. It doesn’t really matter, whether they are the same exact one or slightly modified, what matters is they both yield pretty much the same end result in performance.

> VisualC++ 5.0 & Visual Basic 5.0 & Visual Studio 97
> http://www.microsoft.com/visualc/press/vsentpr.htm
>
> ....
> Performance enhancements
> All the new versions of products in Visual Studio 97 offer faster performance. The performance of existing programs created with Visual C++ will typically increase about 10 percent after they are recompiled using Visual C++ 5.0. Further, Visual Basic 5.0 with its new native compiler now provides performance that is very close to that of Visual C++ in language execution.
> ....

Seeing that the project is compiled into C++ some might argue why not just make it in C++. Quite simple really, in VB you can make in 1 day what takes you 7 to make in C++ and you get pretty much the same results. Couple that with all the support for VB and all of the apps that integrate VBA, it provides the VB programmer with an expandability/integratability previously unheard of. So in reality you get the best of both worlds, the RAD and environment from VB with the speed of C++ all in one language, VB 5.0.

- Dave Sherman

E. What are some other good sources for information?

There are many Web sites out there dedicated to the Visual Basic programmer, Below are a few of the better ones we have found. If you have a VB site that you think should be listed here drop us a /MSG on IRC.

Phaethon's Visual Basic Help Page
http://www.awod.com/gallery/rwav/robertjh/vbasic/

Carl & Gary's Visual Basic Home Page
http://www.apexsc.com/vb

Microsoft's VB Home Page
http://www.microsoft.com/vbasic/

Codd Multimedia Home Page; Home of VB Online
http://www.vbonline.com/

Advanced Visual Basic
http://www.duke-net.com/vb/

Ask the Visual Basic Pro
http://www.inquiry.com/techtips/thevbpro/index.html

The Cobb Visual Basic Website
http://www.cobb.com/ivb/index.htm

Visual Basic Broad-Band Network
http://www.geocities.com/SiliconValley/4548

III. The Voices Of Experience (Tips From The Pros)

A. Dave Sherman

The Future of Programming

The world greatly depends on computers, in the future it will become even more reliant upon them. But how much of a future is there? The physical computer is nothing without the programmers who tell the computer what to do and how to do it. While there is no shortage of programmers there is a vast shortage of creativity and innovativity among them. What's worse is how the term quality seems to have been forgotten in the shuffle, with more and more programmers not paying attention to the details that once allowed them to create quality software. Not only do we risk a future of a standstill on standards but a future where the quality of the programs continues to degrade.

I originally got involved with programming computers because I couldn't resist the possibilities. With life and reality there are many limitations and restrictions. What fascinated me with programming computers is there are no limitations, you are only limited to however far your imagination can take you. You can create things that have never been seen, heard, or even ever thought of before. You can create and do things that would be impossible in real life, from making applications that do things hundreds of times faster then a human could, to creating games that take you to new worlds and places that could only exist in one's mind. It can all be done with computers, and it is all made possible by good programmers.

iD Software made and broke many of the rules of programming and computing when they created Wolfenstine, Doom, and Quake. All defied what people thought could be done and broke many of the standards of their respective times. The innovative programmers at iD Software never let what had/hadn't been done or what other people said could/couldn't be done stop them. They pushed ahead by conceiving new ideas and developing those ideas/dreams into their software thereby making those ideas a reality. For gamers around the world iD has earned themselves a name that very few other companies can compete with. All because they refused to accept the way things were. They decided to take a risk and step into the unknown, designing/developing something that had never been attempted on such a scale before.

Microsoft has done much the same things with their programs. Constantly pushing them to the limits implementing new features and integrating new ideas into them. Over the years we have come from a text based MSDOS to a fully graphical Windows95/NT. They have increased the productivity of computer users around the world by countless amounts. MS has brought the world of computing to the future and even as you read this they continue to bring us even closer to it. This is what separates the good from the great. The great are never satisfied, they can always improve on their work taking it to the next step in computing and pushing their programs ahead to the future.

New Ideas, New Methods – The Push to the Future

Don't get me wrong, standards in programming are there for a reason. But a programmer set those standards and it will be a programmer who steps up to the plate to create the new standards. As all of the creative/innovative programmers decline who will be here tomorrow to progress programming to the next level? If nobody creates the new standards then computer software will go no further. As an example, If iD software hadn't improved on the standards/ideas they set forth with Wolfenstine, Doom and Quake never would have evolved.

Rather then being encouraged to come up with their own ideas and work towards a better way of doing things, most programmers are taught one way of doing things and are warned never to deviate from that allotted method. Maybe it's a flaw in the teaching methods not allowing for individual thought and ideas, or maybe it's a flaw in programmers nowadays looking for the **easy** way of doing things. But no matter what the reason, it is most definitely a problem. Very few programmers and even fewer companies nowadays try to set new standards with their work.

The programming industry needs more programmers that don't accept things the way they are. Programmers that are constantly trying to improve on pre-existing programs, and programmers that will conceive and create totally new programs unlike anything else in the industry. There are very few companies that seem to promote this, Microsoft being one of the only ones that comes to mind. They even run contests promoting creativity, new ideas, and new standards offering prizes to those that can design the best ActiveX controls etc.

Quality Starts at Home

On a more individual basis the industry needs programmers that genuinely care about their work and programs, and that continually find more ways to improve on them. If each programmer took a greater interest in the quality of their work, the programming industry would be a better place. There is a saying shared by many cultures from around the world that talks about how a part of your life, a part of your essence, and a part of you is kept in everything you make. Each of us puts of ourselves into the different projects/things we create in our lifetime. If you don't do the best you can on a project it does nothing but reflect badly upon you. The legacy of many of the programs and standards we create now will be around long after we are gone, so how do you wish to be remembered?

There are many things that make a program into a quality program. From a good/easy to use user interface, to optimizing your code to work at peak performance, to using some of the lesser known trade secrets, to providing a lot of functionality in your program, all the way down to creating an easy to understand yet comprehensive on-line help system. Many programmers seem to neglect these nowadays. For instance a lot of programmers think performance must be sacrificed to provide more functionality. However, many programs offer a full range of functions without sacrificing performance. A good example here is to compare the performance of MS IE 3.0 vs. Netscape 3.0, both offer pretty much the same options but IE is considerably faster then Netscape. If you optimize your programs correctly you can add more functionality with little to no performance drop.

The key here is to take the time needed to produce the quality software. Rushing things does nothing but create cut corners and programs of lesser quality. With the ever growing time constraints being placed on programmers to get the job done, more and more programmers are sacrificing the high quality of their work just to get the job done faster. This just shouldn't be done. If it can't be completed in the estimated time the last thing you should ever do is sacrifice the quality of the work or cut corners. I know of many companies that do this and the thing I always hear back is how sorry they are they did it. People pay for the software thinking it will be of the high-quality that they have come to expect form that company and get nothing but disappointments. In the end what happens is as the word gets out that the software isn't' as high quality as previous versions the sales drop off and the companies name/reputation becomes tarnished. It's better to take an extra week or to give slightly larger estimates of how long a project will take to make sure that the quality is still there. As a rule of thumb, if you are not 100% satisfied with your work or you are not proud to have your name on the project, it probably isn't fit to be released.

Final Thoughts

So in the end, my best tip is to never stop looking for new and better ways to do things. Just because you hear or read something telling you how to do things or what can or can't be done. Don't accept it. There are many ways to accomplish that which you want to do. Programming methods, when designed, are integrated to do a certain function the development team had in mind. However, many times the advanced programmer can find ways to use those very same methods to do things they were never intended to be able to do.

Keep the quality software coming, if you aren't overjoyed with it's performance and appearance, then it's not done to the level of quality it should be. Never be afraid to take your time. A company would much rather have software that performs and works better then the competition then have shoddy software that they then have to fix later on. Always overestimate how much time a project will take, this will give you some extra breathing room should something go wrong or the quality standard not be met. Always try to meet your deadlines, just not at the cost of the quality of your programs. If your project runs over the estimated time you may wish to take yourself off the clock so that the company paying for it isn't getting stuck with the bill.

To the beginners, don't jump in head first. You don't jump into the deep end of the pool without knowing how to swim. Start with something easy and work your way up to more complicated programs. A good program to start with is a text editor like Notepad, then gradually work your way up. Always keep backups of your work in many different secure places. Having a HD sector go bad or a HD crash can be quite detrimental to a project if it's not backed up. Also keep archives of your backups so you can go back and unarchive the backups form a couple days ago if you need to.

I Wish You Success In Your Programming Endeavors,
Dave Sherman

B. Robert Houghtaling

Thoughts, Guidelines and Recommendations

When I sit down to write a program, be it a large and complex design or be it a test app to see if I can blit to the screen a little faster, I feel very Powerful. Like I can do anything, because the possibilities are infinite. There is a little excitement in me that knows anything can happen in a program and I look forward to it every day. I hope you can find that same enthusiasm in your work.

Over the years I have found some things which might help you. They aren’t rules, like Microsoft’s design guide for developing Windows applications is; they’re simply a few of the things I have found to make my life easier. They allow me to be a better developer and help to create better and more stable applications.

Learn the VB Environment

Learn its advantages, quirks and features before you delve into any sort of serious programming. This becomes a matter of learning what you can do before figuring out what you want to do. Look at every menu item, hotkey, dockable/non-dockable window, toolbar button and dialog option. If you see what VB does with itself by necessity, it will help you keep clear in your mind what is going on inside your program. If one part of the compiler does not make sense to you, go right ahead and skip to other things; (as long as you remember to come back to it later) it might make more sense after you’ve looked at some other aspect of VB (or after a good cup of coffee).

Bad Color Schemes

It is very tempting to try to come up with the "perfect" color scheme when writing a program, but please stop using yellow and red as your form background. Use the VB color constants (i.e. vbButtonFace, vb3dHighlight, etc.) to "paint" objects on your forms. If you really feel the need to use custom dialogs and "build a better mousetrap" for the user-interface, keep in mind that those type of forms and applications are usually very confusing to the user. The consistent interface of Mircrosoft Windows-based programs is one of the primary advantages that drastically shorten the learning-curve of an application for the user. Sure, it can be interesting, but does the background of your form really need to look like a CGI-rendered molded plastic day-glo orange safety vest? (see: Kai’s "Confusing" Power Tools for an example of an application which uses those custom-generated "owner-drawn" forms as the GUI. It is very pretty, but many people find it ultimately confusing and difficult to use.)

Variables, Variables, Variables

Always dimension your variables and use the Option Explicit directive with the "Require Variable Declarations" user-option. At first this may seem confusing as "normal" BASIC doesn't require this, so why should you be bothered with such a feature in Visual Basic. Well, it turns out that the inherent typos and confusing names combined with the large complexity of most programs can easily create major bugs in your program. Spelling lngConversion as lngCoversion someplace deep within your application and NOT using Option Explicit might work 99% of the time because of the conditions at that point, but that 1% of exceptions can be devastating. I have seen programmers who say they never use Option Explicit and say they never have a problem, but in my experience its one of the nicest features in a compiler that I’ve ever used. I ask why one would not bother with a feature that can do nothing for you except save time and find bugs.

Debugging in Visual Basic

Learn how to use VB’s debugging capabilities. It has been said (and I tend to agree) that VB has the most powerful (run-time) debugger of any of the higher-level languages. Breakpoints, stepping, constant watching of the context of variables and current statement setting are just some of the rich features of VB's run-time debugger. I seriously recommend learning to debug your apps. MsgBoxes to see variables at run-time are not usually necessary if you can learn to use those features.

Conditional Compiling

Conditional compilation is a feature of VB introduced with version 4.0 that has many uses, which range from: writing platform-independant code, compiling in and out features, and debugging an application. Its difficult to recommend any one given method of conditional compiling to a programmer, as its dependant on what you want to do. The only undocumented warning I have (at least, I believe that it is still undocumented) is that the conditional arguments textbox in Tools|Options requires a : (colon) to separate multiple items:

```
ARG_ONE=1 : ARG_TWO=2
```

Design Conclusion

Lastly, I recommend looking around. Look around at other professional applications that have been created both by outside developers and by Microsoft. Believe it or not, neither Microsoft nor all of the Windows-platform developers know everything, and you (we all) can take advantage of it and write incredible applications. Every day, someone, somewhere is writing the next word processor, web browser, memory manager or image viewer (and yes, somebody is writing another IRC client) and they are using techniques that no one else considered. Original ideas are the "soul" of programming.

Use Microsoft Word and Word Perfect as "models" of good (or not-horrible) programming design when you are looking to write something that might be a document editor. Look at Internet Explorer and Netscape Navigator when wanting to write something like a web browser. I see many, many people wanting to write applications, yet seem to know nothing of what they want their program to do or how they want to go about doing it. (See: the GIF Construction Set by Alchemy Mindworks and then compare it to Microsoft’s GIF editor and see which one you like better. My preference in this case is for Microsoft because A.M. put little or no thought into the design of their program and it feels clunky and generally useless because of its inefficient dialogs and sloppy design, (don’t you despise incorrect tab-ordering?) even though it is slightly more feature rich than Microsoft’s editor.)

Judge what you need your program to do and then figure out how to do it. Please do not misunderstand me--I am not condoning plagiarism, but do not be afraid to use ideas you get from looking at someone else’s program. They say form follows function and seeing other designs and techniques can inspire ideas and thoughts and should lead to improvements, cleaner design and generally better applications.

Robert Houghtaling

C. Charles Haeberle

One of the finest books on programming I've ever read was not related to any specific programming language. The book, "Code Complete" (Steve McConnell, (c) Microsoft Press, 1993) is "A Practical Handbook of Software Construction", and details problems and solutions in the programming process. The book focuses on the actual construction phase, which partly encompasses design and testing, but primarily focuses on Coding and Debugging. This is a book which ***every*** programmer should read. And while I certainly can't replace his book, nor would I even try, I will attempt to discuss one of points which has had the greatest impact on my programming accuracy and performance. So I urge you to consider this concept in your programming, and to read Mr. McConnell's book as soon as possible!

Procedure Definition Language

Programmers are doers. We like to get in there and make things happen. However, we too often don't think things through before we dive right into code. Then we get halfway into a complex procedure and find ourselves with an overly complex solution to what we thought was a simple problem. Why does this happen? Because, to use a potentially painful metaphor, we dove into the pool and found out too late which end we were at. Even worse, we whip out a function, and then a month or so later we need to go and make a change, and can't remember how it works because we forget (in our rush to make things happen) to comment our code.

Fortunately, there's a very helpful and easy to use technique to help avoid both problems. That technique is PDL - Procedure Definition Language. PDL is not written in any computer based language, but in clear English. Its a step by step listing of all the things a procedure needs to accomplish, in order, including decision branches and calls to other procedures. In essence, its writing out comments before you write a word of code. All too often we get halfway into a process before we realize we've got to go back and change what we've already written. Every programmer gets attached to their code, and having to go back and change it after we've invested "blood sweat and tears" into it hurts. By using PDL, we write out simple steps in English which can be modified or rewritten very easily before we invest a single line of code into the procedure. In addition, it helps early on to identify areas of a procedure which really belong in their own function. The ultimate result, then, is increased efficiency due to better planning, well commented functions, and faster production time because of heightened accuracy writing the code the first time.

Building a PDL procedure is done in two parts. The first part should be a comment block which identifies the purpose of the procedure being written, and a general concept of the flow of that same process. The second part is to expand that comment block into a series of step by step instructions. These instructions should be specific enough to outline specific steps which need to be taken in the procedure. This isn't to say that there should be as many lines of PDL as there ultimately are of code, of course. However, the more granular the PDL, the better comments as well as the planning, and the more likely that algorithm-bug free code will be the initial product.

Below is an example from an application I'm currently working on. This particular section of code controls the startup of the application. The first section is the PDL which was written prior to the actual construction of the code. The second section shows the actual completed code.

```
Private Sub Class_Initialize()
        ' This routine performs the actual application startup.  All necessary connections
        ' are initialized, object structures created, etc.  frmSplash.SetStatus is used to
        ' keep the user informed of the current activity and show progress.  (Perception of
        ' action makes the application appear faster, even though the actual
        ' implementation slows it down slightly.)

        ' Set hourglass
        ' Initializes application and all child objects
        ' Locate the application options storage area.  (INI file in 16 bit, registry in 32 bit)
        ' Bring up splash screen
        ' Connect to data sources - if it fails, exit
        ' Once connected, get local user information such as username, level, etc.
        ' Load the mdi frame into memory
        ' Prep account structure
        ' Prepare starting screen - list of accounts
        ' Load external component references
        ' Display the MDI Frame and list
        ' Remove splash screen and hourglass
End Sub

Private Sub Class_Initialize()
        ' This routine performs the actual application startup.  All necessary connections
        ‘ are initialized, object structures created, etc.  frmSplash.SetStatus is used to
        ‘ keep the user informed of the current activity and show progress.  (Perception of
        ‘ action makes the application appear faster, even though the actual
        ‘ implementation slows it down slightly.)

        Dim mp As Integer
        Dim DbName$, UserName$, Password$, ODBCConnect$
        Dim TempRS As Recordset, SQL$

        ' Set hourglass
        mp = Screen.MousePointer
        Screen.MousePointer = vbHourglass

        WriteDebugFile "Initialization", "Application.Initialize"
        ' Initializes application and all child objects

        'Locate the application options storage area.  (INI file in 16 bit, registry in 32 bit)
        WriteDebugFile "Access initialization parameters", "Application.Initialize"
        #If Win32 Then
                gINIFile$ = App.Name
        #Else
                gINIFile$ = App.Path & "\UDSS.INI"
        #End If

        ' Bring up splash screen
        frmSplash.Display
        frmSplash.SetStatus "Connecting to Database", 5

        ' Connect to data sources - if it fails, exit9
        If Not ConnectDatabases Then ' This process updates the splash screen to 70
                frmSplash.Quit
                Me.Quit
                Exit Sub
        End If

        ' Once connected, get local user information such as username, level, etc.
        SQL$ = "SELECT USER_FIRST_NAME, USER_LAST_NAME, _
        USER_UDSS_LVL FROM "
        SQL$ = SQL$ & DBCreator & ".CATUW12_USER WHERE _
        USER_UDSS_ID = '" & gCurrentUserID$ & "'"
        Set TempRS = OpenBackEndRecordset(SQL$)
        gCurrentUserName$ = Trim$(TempRS.Fields("USER_FIRST_NAME")) _
        & " " & Trim$(TempRS.Fields("USER_LAST_NAME"))
        gCurrentUserLevel% = TempRS.Fields("USER_UDSS_LVL")
        Set TempRS = Nothing

        ' Load the mdi frame into memory
        frmSplash.SetStatus "Initializing MDIFrame", 75
        MDIFrame.Hide

        ' Prep account structure
        frmSplash.SetStatus "Initializing Object Tree", 80
        Set Me.Accounts = New Accounts

        ' Prepare starting screen - list of accounts
        frmSplash.SetStatus "Preparing Account List", 90
        frmAccounts.Init

        ' Load external component references
        frmSplash.SetStatus "Locating Components", 95
        GetComponents

        ' Display the MDI Frame and list
        frmSplash.SetStatus "Underwriting DSS Initialized", 100
        MDIFrame.Display
        frmAccounts.Show

        ' Remove splash screen and hourglass
        frmSplash.Quit
        Screen.MousePointer = mp

    End Sub
```

Hopefully, by comparing these two examples, you'll see how building the flow of the program into the comments first helps planning and efficiency. If you're not convinced, consider this: ***I have never had to debug this procedure.*** In fact, the most I ever had to do was add a line or two to the "get user info" section when we added more information. By writing the PDL first, I had a map telling me where to call other functions, when to do logical tests, and so on. Its when I don't spend enough time in PDL that I wind up with code that gets tweaked and tweaked again.

One final point. If you fail to comment and properly plan out your code, then one day someone may come along and develop a strong desire to find you and punish you for your sins. If you're a solo-programmer, that person may very well be you. I can't tell you how many times I've gone back to a program I wrote a few months earlier and been totally lost. By use of PDL, I make code maintenance much easier, and the long-term savings of easy to maintain code should be self-evident.

This is by far not the only area where programmers tend to fall short. If you are serious about developing not only good coding skills, but also good programming methodology, you will get Steve McConnell's book.

See you on-line!
Charles Haeberle

IV. Common Questions/Issues With Visual Basic

A. Working With Controls

1. How do I save and read to textboxes?

Reading And writing text files requires use of file i/o functions. The functions/statements we will be using are the OPEN, CLOSE, LINE INPUT, EOF, and PRINT. First create a form that has one textbox and 2 command buttons on it. Make sure the textbox has it's MultiLine property set to true. Below is the code to place in the control events as shown. We will be using the first command button to save the contents of the textbox to a file called "MyFile.txt" and we will be using the second button to load the text from the file. Please note the use of the "#1" in all statements involving the file. We use this file handler to identify which file we are talking about. At a given time you program could have a lot of files open for either reading or writing and this provides us with a more structured way to keep track of them and to identify them.

```
Private Sub Command1_Click()
        Open "MyFile.txt" For Output As #1
                ' This opens the file "MyFile.txt" in the current directory for output.
                Print #1, Text1.Text
                ' This prints all of the text in the textbox to the file.
        Close #1 ' This closes the file we just opened.
End Sub

Private Sub Command2_Click()
        Text1.Text = "" ' Clear the textbox
        Open "MyFile.txt" For Input As #1 ' Open "MyFile.txt" so we can read it
                Do Until EOF(1)
                        ' Do the following code until we reach the end of the file
                        Line Input #1, temp$ ' Read in a line from the file
                        If Text1.Text <> "" Then
                                ' If the textbox has text in it add the new text to it
                                Text1.Text = Text1.Text & vbCrLf & temp$
                        Else
                                ' If the textbox is empty just stick our new text in it
                                Text1.Text = temp$
                        End If
                Loop ' Loop back to the do as long as we aren't at the end of the file
        Close #1 ' Close the file
End Sub
```

Ok, now that you have all that code added run the program. Type a few lines of giberish and click the first button to save it. Now change the textbox so you'll notice a difference when you reload the text file. Click the second button and you are back to your original text that you saved. Also if you check your vb directory you'll see a text file called "MyFile.txt", that's the file you just wrote. There is a lot more you can do with VB's file i/o but this is a good start to get you along the right direction.

- Dave Sherman

2. How do I stop the textbox from beeping when I press enter?

There are a couple different ways to solve this problem, and the method to choose is really dependent on what effect you want to accomplish. Choose what you want form the options below.

One-line textbox that doesn't do anything when the user presses enter.

```
Sub Text1_KeyPress (KeyAscii As Integer)
    If KeyAscii = 13 Then KeyAscii = 0
End Sub
```

One-line textbox, and a command button on my form to execute when the user presses enter.

Set the command button's "Default" property to true. This will cause the command button to take the enter key as a click on itself, and the textbox will not beep.

A textbox that moves the cursor down to the next line when enter is pressed.

Set the textbox's "MultiLine" property to true. This will make the textbox like the windows notepad editing window.

- Dave Sherman

3. How do I use multiple colors in a textbox?

A normal textbox can not support multiple colors. So if you wish to have more then one color in your textbox at a time you have one of two options, use a RichTextBox or use a PictureBox. Below is a small example to get you started for each showing how to add text to them in different colors. Create a form with 2 command buttons, a RichTextBox, and a PictureBox on it. Make sure you keep all standard names or you will have to change the code to reflect your object name changes.

```
Sub Command1_Click()
    RichTextBox1.Text = “”
    For x = 0 to 15
        RichTextBox1.SelColor = QBColor(x)
        ‘ set the current color ( you don’t ‘need to use the 16 color qb palette
        ‘ I’m using, I’m just using it to simplify color selection. )
        RichTextBox1.SelText = "Color" & x & vbcrlf
        ‘ add our new text
    Next x
Exit Sub

Sub Command2_Click()
    Picture1.Cls ‘ Clears the PictureBox
    For x = 0 To 15
        Picture1.ForeColor = QBColor(x) ‘ sets the current color
        Picture1.Print "Color" & x ‘ add our new text
    Next x
End Sub
```

- Dave Sherman

4. How do you set tabs in a listbox?

A simple example, in a module declare:

```
Public Const LB_SETTABSTOPS = &H192
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd _
As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Long) As _
Long
```

Then in the sub or function where you are adding to the list box:

```
Dim rc As Long
Dim tabStops As Long
tabStops = 20
rc = SendMessage(List1.hwnd, LB_SETTABSTOPS, 1, tabStops)
List1.AddItem "This is" & vbTab & "tabbed"
```

For multiple tabs, make tabStops an array, and set the third (wParam) parameter to SendMessage() to the number of items in this array.

- Guy Thompson

5. How do I find a particular word in a textbox and change it?

Create a new project, on Form1 place a textbox and a command button. Now add the events below. This example will add a phrase to the textbox then will search/replace the word “Peter” with “Bob” when you click the command button

```
Private Sub Form_Load()
    Text1.Text = "Peter Piper picked a peck of pickled peppers. If Peter Piper _
    picked a peck of pickled peppers then how many pickled peppers did Peter _
    Piper pick."
End Sub

Private Sub Command1_Click()

    Dim SearchText As String
    Dim FoundPos As Long
    Dim ReplaceText As String
    Dim EndPos As Long

    FoundPos = 1
    SearchText = "Peter"
    ReplaceText = "Bob"

    Do
        FoundPos = InStr(FoundPos, Text1.Text, SearchText)
        If FoundPos <= 0 Then
            Exit Sub
        End If
        EndPos = InStr(FoundPos, Text1.Text, " ")
        Text1.SelStart = FoundPos – 1
        Text1.SelLength = EndPos – FoundPos
        Text1.SelText = ReplaceText
    Loop While InStr(FoundPos, Text1.Text, SearchText) > 0
End Sub
```

- Dave Sherman

6. How do I scroll a textbox like the mIRC channel window?

Your textbox must have the MultiLine property set to true, and the ScrollBars property set to 2 (show vertical). Use the following in your textbox's change event.

```
Private Sub MyTextBox_Change()
        MyTextBox.SelStart = Len(MyTextBox.Text)
End Sub
```

This causes the textbox to scroll to the bottom of the textbox every time the contents of the box changes. There is a somewhat negative sideaffect, however, in that you might be scrolling upwards when text is added, and this code will cause the textbox to jump to the end again. At this time, I know of no way to prevent this side affect, although I am sure there is at least one way to block it.
- Charles Haeberle

7. How do I add a VBX/OCX to my project?

VB 5.0 - Select Project->Components from the menu bar for a list.
VB 4.0 - Select Tools->Custom Controls from the menu bar for a list.
VB 3.0 - Select File->Add File from the menu bar, and browse \windows\system for the file you want
- Guy Thompson

8. How do I make something happen at regular intervals?

We will separate this into 3 different intervals. Intervals under 65 seconds apart , intervals over 65 seconds apart, and intervals that occur a certain time during the day. All of these examples use the timer control, just in different ways to accomplish the different effects.

If you wish your procedure to fire every 65 seconds or less, set the timer control's interval property to the desired number of milliseconds.

60000ms = 60 seconds = 1 minute
30000ms = 30 seconds
01000ms = 01 second

Now in the timer's event either place your procedure you wish to execute every x amount of seconds or call your procedure from the timer event using the call statement.

If you wish your event to fire at a time greater then 65 seconds you have t do a little fancy footwork. The timer control only allows intervals of a little over 65 seconds. So what you have to do is break up your desired interval into smaller chunks. If you want 100 seconds you could split it up into two 50 ms events, five 20 second events, or ten 10 second events. To keep track of the actual number of times the event has fired you will have to declare a variable in the form's declare section.

In this example we want our event to fire every five minutes. Create a new project, put a timer control on form1 and set the timer controls interval property to 60000ms.

Form's Declare Section:

```
Dim MyTimer As Long ' Var keeps track of how many times the event has fired.

Private Sub Timer1_Timer()
        MyTimer = MyTimer + 1 ' Add to the fire var so we know the event has fired
        If MyTimer >= 5 Then ' Check to see if this is the 5th time it's fired (5 mins)
                MyTimer = 0 ' Reset the var
                Call MyProc ' Call the procedure we wish to execute
        End If
End Sub
```

Finally, let’s say you wished your event to fire once a day at a given time. In this example set the timer’s interval property to 60 seconds. For this example we will fire the event at 2:25am.

```
Private Sub Timer1_Timer()
        If Hour(Now) = 2 And Minute(Now) = 25 Then
                ‘ Compare the time against what we are looking for
                Timer1.Enabled = False
                Call MyProc ‘ Call the procedure we wish to execute
                Timer1.Enabled = True
        End If
End Sub
```

- Dave Sherman

9. How do you add/remove items from a listbox?

First thing to remember when using listboxes is that they are zero based. What this means is the items start with an index of 0. So if there were 5 items in the list box they would have indexs of 0-4. If you do not specify an index value when adding items they default to being added onto the end of the list.

To add an item to a listbox use the AddItem method

```
List1.Clear ‘ Clears the listbox
List1.AddItem ("Foo") ‘ Adds item “Foo” to the listbox at index 0
List1.AddItem ("Foo2") ‘ Adds item “Foo2” to the listbox at index 1

List1.AddItem ("Foo3", 0) ‘ Adds item “Foo3” to the listbox at index 0
```

In the listbox you will see the items listed in the order of Foo3, Foo, Foo2. Because we specified an index value of zero for Foo3 it was placed in that index and whatever item, Foo in this case, had that value before it will be moved down on the list directly after the newly placed item.

In order to remove an item in a listbox you must first know it’s index. If you know it’s index you can just do a simple.

```
List1.RemoveItem 0 ‘ Remove item with the index of 0
```

The above code will remove whatever item has the index of 0. All other items will then shuffle up, item 1 will become 0, item 2 will become 1 and so forth. But what happens if you don’t know an item’s index you just know the data in it.

```
List1.Clear ‘ Clear the listbox
For x = 0 To 8
        List1.AddItem ("Foo" & 0) ‘ Add some items to the listbox
Next x
For x = 0 To List1.ListCount – 1 ‘ Loop through list
        ‘ remember index 0 counts as an item so the ListCount will always be one
        ‘ higher then the top boundary. If you have 3 items they are 0-2 not 1-3
        If List1.List(x) = "Foo6" Then ‘ Check this items data
                List1.RemoveItem x ‘ Data matches, so remove the item
                Exit Sub ‘ Exit the procedure because we found what we wanted
        End If
Next x
```

If at all possible try to keep track of the item data/index of that data internally in your program rather then looping through to find what item contains what data.

- Dave Sherman

10. How do you add/remove items from a combobox?

To add to a combo box:

```
Combo1.AddItem "Item"
```

To remove from a combo box:

```
Combo1.RemoveItem Index   (0 based)
```

- Guy Thompson

11. How do you add/remove items from a listview control?

```
Dim lv As ListItem
Set lv = ListView1.ListItems.Add(, , "item " & 1)
```

- Guy Thompson

12. How do I turn on full-row-select in a listview control?

The ListView and other common controls found in the COMCTL32.OCX file only come with VB4-32pro or better, so people with VB4-32std and below can safely ignore this one.

By default the ListView control only highlights the word you choose, like in the Explorer's file view. However, many other MS apps use the same ListView and allow the entire line to be highlighted. You turn on/off this ability by using the SendMessage() API function to set an extended style: LVS_FULLROWSELECT

***** 32-bit Sample *****
In a project with 1 ListView, place the following code:

```
Option Explicit

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
(ByVal hwnd As Long, ByVal Msg As Long, ByVal wParam As Long, ByVal _
lParam As Long) As Long

Private Const LVM_FIRST = &H1000
Private Const LVM_SETEXTENDEDLISTVIEWSTYLE = LVM_FIRST + 54
Private Const LVM_GETEXTENDEDLISTVIEWSTYLE = LVM_FIRST + 55
Private Const LVS_FULLROWSELECT = &H20

Private Sub Form_Load()
        Dim itmX As ListItem

        Set itmX = ListView1.ListItems.Add(, , "Sample Person 1")
        itmX.SubItems(1) = "29"
        itmX.SubItems(2) = "Male"
        Set itmX = ListView1.ListItems.Add(, , "Sample Person 2")
        itmX.SubItems(1) = "29"
        itmX.SubItems(2) = "Female"
        Set itmX = ListView1.ListItems.Add(, , "Sample Person 3")
        itmX.SubItems(1) = "26"
        itmX.SubItems(2) = "Male"

        ListViewFullSelect ListView1, True
        ListView1.Refresh
End Sub
```

```
Private Sub ListViewFullSelect(pcList As Control, bTurnOn As Boolean)
        Dim lStyle As Long
        Dim lResult As Long

        lStyle = SendMessage(pcList.hwnd, _
        LVM_GETEXTENDEDLISTVIEWSTYLE, 0, 0)

        If bTurnOn Then
                lStyle = lStyle Or LVS_FULLROWSELECT
        Else
                lStyle = lStyle And (Not LVS_FULLROWSELECT)
        End If

        lResult = SendMessage(pcList.hwnd, _
        LVM_SETEXTENDEDLISTVIEWSTYLE, 0, lStyle)
End Sub
```

- Robert Houghtaling

13. How do you create controls at run-time?

Visual Basic allows creation and destruction of controls at run-time by means of a control array. As the developer, you must create one control at design time which is a 'model' for the control array. Every control in the array will behave exactly the same as the model, but can have different properties. For example, if you create a picturebox control array and create multiple instances of the picturebox at runtime, each picturebox can be in a different location on the form, have a different picture loaded, etc. However, if you design a click event for that picture box which lets the user select a particular picture to be loaded, every instance of the picture box will have the same functionality at run-time.

To create the 'model' control, add the control to your form as you would normally, but set the Index property of the control to 0. The Index property (which is normally null) identifies the control as being part of a control array. The index property can be any integer. It is the existence of an index, not the specific value, which defines it as an arrayed control. The index also identifies each unique instance of the control at run-time, as you will see.

When you are using a control array, all methods and events for your control receive an additional parameter, Index. This parameter is used to identify exactly which instance of the control received the event or is being asked to perform a task. When using a control array, you refer to it in the same way you refer to individual elements in a normal array, by placing the index number in parenthesis after the name of the array. MyControl(1), MyControl(2), etc.

At run time, you must track or determine what index values are available before you can load a new instance of a control. If your model control is Index 0, you can not create a second control with Index 0. One method of tracking is to load multiple instances sequentially, and to use a global variable to identify the last index number loaded.

Ok, enough theory. To try out loading and unloading control arrays, try out this example from the Visual Basic help file. (Visual Basic 5.0 Help file topic "Index Property (Control Array)" Example)

This example starts with two OptionButton controls and adds a new OptionButton to the form each time you click a CommandButton control. When you click an OptionButton, the FillStyle property is set and a new circle is drawn. To try this example, paste the code into the Declarations section of a form that has two OptionButton controls, a CommandButton, and a large PictureBox control. Set the Name property of both OptionButton controls to optButton to create a control array.

```
Private Sub OptButton_Click (Index As Integer)
        Dim H, W        ' Declare variables.
        Picture1.Cls        ' Clear picture.
        Picture1.FillStyle = Index        ' Set FillStyle.
        W = Picture1.ScaleWidth / 2        ' Get size of circle.
        H = Picture1.ScaleHeight / 2
        Picture1.Circle (W, H), W / 2        ' Draw circle.
End Sub

Private Sub Command1_Click ()
        Static MaxIdx    ' Largest index in array.
        If MaxIdx = 0 Then MaxIdx = 1 ' Preset MaxIdx.
        MaxIdx = MaxIdx + 1    ' Increment index.
        If MaxIdx > 7 Then Exit Sub        ' Put eight buttons on form.
        Load OptButton(MaxIdx)        ' Create new item in array.
        ' Set location of new option button under previous button.
        OptButton(MaxIdx).Top = OptButton(MaxIdx - 1).Top + 360
        OptButton(MaxIdx).Visible = True        ' Make new button visible.
End Sub
```

- Charles Haeberle

B. Working With Files

1. How do I find out how big a file is?

The FileLen function returns the size of a file in bytes.

```
Result = FileLen("C:\AUTOEXEC.BAT")
```

- Charles Haeberle

2. How do I delete a file?

The Kill function deletes the file specified.

```
Kill "C:\MyFile.Txt"
```

- Charles Haeberle

3. How do I rename a file?

The Name function renames a file.

```
Name "C:\OldFile.Txt" as "C:\NewFile.Txt"
```

- Charles Haeberle

4. How do I copy a file?

To copy a file use the FileCopy functions as shown below.

```
FileCopy "C:\File.Txt", "C:\FileCopy.Txt"
```

- Dave Sherman

5. How do I move a file?

Moving a file involved two functions. First copying the file to a new location, then deleting the old file.

```
FileCopy “C:\File.Txt”, “C:\FileCopy.Txt” ‘ Copy file to a new file
Kill “C:\File.Txt” ‘ delete the old file
```

- Dave Sherman

6. How do I check to see if a file exists?

Checking to see if a file exists is actually quite easy. All that it really entails is using the Dir$ function to check for a file pattern, then check the result to see if it’s the file you want.

In the example below the pattern doesn’t have any wildcards because we know exactly what file we are looking for. As you see I also used the LCase$ function because OPTION COMPARE TEXT might not be on, in which case the capitalization of the saved file would make quite the difference. When changing this code to meet the requirements of your program make sure you keep the filename in lowercase for comparison reasons.

```
If LCase$(Dir$("c:\autoexec.bat")) = "autoexec.bat" Then
        ' File Exists
Else
        ' File Doesn't Exist
End If
```

For the more advanced programmer, if your program checks multiple files when it runs, you may want to create a function with the above code in it and call it something like FileExists. Have the function set up to receive a filename as an argument, check if the file exists, and return a numerical value that would tell the calling procedure if it does or not. Then just evaluate that return value. In the end it will save you a lot of duplicated code which will in turn slightly increase overall performance.

- Dave Sherman

C. Configuration

1. How do I read/write INI Files?

For VB3, VB4-32 and VB5, there are the two API functions, Get- and WritePrivateProfileString().

For VB4-16 you can use those API functions or you can use the built-in VB functions Get- and SaveSettings.

An INI file is simply a text file that is formatted in a certain way. There are three parts: application, key and value

```
[ApplicationName]
Key=Value
```

***** 16-bit Sample *****
In a project with 1 commandbutton, place the following code:

```
Declare Function GetPrivateProfileString Lib "kernel" (ByVal lpApplicationName _
As String, ByVal lpKeyName As String, ByVal lpDefault As String, ByVal _
lpReturnedString As String, ByVal nSize As integer, ByVal lpFileName As String) _
As integer
Declare Function WritePrivateProfileString Lib "kernel" (ByVal _
lpApplicationName As String, ByVal lpKeyName As String, ByVal lpString As _
String, ByVal lpFileName As String) As integer
```

```
Sub Command1_Click()
        Dim sData As String
        Dim iDataLen As integer

        ' Read from WIN.INI
        ' [Desktop]
        ' Wallpaper={path\filename.bmp}

        sData = Space$(255) ' Allocate space for the string
        iDataLen = GetPrivateProfileString("Desktop", "Wallpaper", "", sData, _
        Len(sData), "win.ini")

        ' strip off the NULL that API routines put on the end of strings
        sData = Left$(sData, iDataLen)

        ' Print the string to the debug window
        Debug.Print sData

        ' Now write to WIN.INI
        ' [Desktop]
        ' Test=here is some text
        sData = "here is some text"
        WritePrivateProfileString "Desktop", "Test", sData, "win.ini"
End Sub
```

***** 32-bit Sample *****
In a project with 1 commandbutton, place the following code:

```
Option Explicit
Private Declare Function GetPrivateProfileString Lib "kernel32" Alias _
"GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal _
lpKeyName As String, ByVal lpDefault As String, ByVal lpReturnedString As _
String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Private Declare Function WritePrivateProfileString Lib "kernel32" Alias _
"WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal _
lpKeyName As String, ByVal lpString As String, ByVal lpFileName As String) As _
Long

Private Sub Command1_Click()
        Dim sData As String
        Dim lDataLen As Long

        ' Read from WIN.INI
        ' [Desktop]
        ' Wallpaper={path\filename.bmp}

        sData = Space$(255) ' Allocate space for the string
        lDataLen = GetPrivateProfileString("Desktop", "Wallpaper", "", sData, _
        Len(sData), "win.ini")

        ' strip off the NULL that API routines put on the end of strings
        sData = Left$(sData, lDataLen)

        ' Print the string to the debug window
        Debug.Print sData
        ' Now write to WIN.INI
        ' [Desktop]
        ' Test=here is some text
        sData = "here is some text"
        WritePrivateProfileString "Desktop", "Test", sData, "win.ini"
End Sub
```

Since Microsoft provides a clean interface to reading INI files, most people will use them...however now I'll discuss a method of how to do it without the API.

Paste the following code block into a form with one CommandButton on it. When pressed, the CommandButton will read the entire contents of an ini file into an array and then search for the requested application and key. If found, it will print the value in the debug window; otherwise it will print the given default value.

This is intended as an example to show methods of reading text files and general string manipulation. If you were to write an internal structure to handle this you would want to use a slightly different method to achieve optimum results.

There is no error handling here...In real life, you will run into improperly formatted INI files, problems opening/reading/writing/closing files, etc. so you should examine each step for possible problems.

```
Private Sub Command1_Click()
        Debug.Print MyGetINI("intl", "sCountry", "none", "c:\windows\win.ini")
End Sub

Private Function MyGetINI(ByVal Application As String, ByVal Key As String, _
ByVal Default As String, ByVal lpFile As String) As Variant

        Dim hFile As Integer
        Dim i As Integer, pos As Integer
        Dim sData() As String
        Dim sTemp As String
        Dim bSectionFound As Boolean
        Dim bKeyFound As Boolean

        hFile = FreeFile ' allocate a file handle

        ' Open the file
        Open lpFile For Input As hFile

        ' Loop through the file, ignoring blank lines.
        ' when done, all data will be in the sData() array.
        Do While Not EOF(hFile)
                Line Input #hFile, sTemp

                If Len(Trim$(sTemp)) > 0 Then
                        ReDim Preserve sData(0 To i) As String
                        sData(i) = Trim$(sTemp)
                        i = i + 1
                End If
        Loop
        ' Start the search for the correct application.
        ' UCase$() is used here to be sure that upper and lower casse does
        '   not interfere with our search.
        For i = 0 To UBound(sData)
                If UCase$(sData(i)) = "[" & UCase$(Application) & "]" Then
                        bSectionFound = True
                        Exit For ' we found the application, so no need to continue.
                End If
        Next i

        ' Now we loop until we either find the key, hit another INI application
        '   section or reach the end of the array.
        If bSectionFound Then
                Do
                        i = i + 1
                        ' Look for the = separator.
                        pos = InStr(sData(i), "=")
                        If UCase$(Key) = UCase$(Left$(sData(i), pos - 1)) Then
                                bKeyFound = True
                                Exit Do
                        End If
                        If i = UBound(sData) Then Exit Do ' no more data, so quit.
                Loop While (Left$(sData(i + 1), 1) <> "[")
        End If

        ' If we found the key, we trim off the value and return it
        '    Otherwise, we return the default value.
        If bKeyFound Then
                MyGetINI = Mid$(sData(i), pos + 1)
        Else
                MyGetINI = Default
        End If
End Function
```

- Robert Houghtaling

2. How do I read/write to the System Registry?

“The registry stores data in a hierarchically structured tree. Each node in the tree is called a *key*. Each key can contain both *subkeys* and data entries called *values*. Sometimes, the presence of a key is all the data that an application requires; other times, an application opens a key and uses the values associated with the key. A key can have any number of values, and the values can be in any form.

Each key has a name consisting of one or more printable ANSI characters — that is, characters ranging from values 32 through 127. Key names cannot include a space, a backslash (\), or a wildcard character (* or ?). Key names beginning with a period (.) are reserved. The name of each subkey is unique with respect to the key that is immediately above it in the hierarchy. Key names are not localized into other languages, although values may be.” --*Microsoft Win32 SDK*

To read and write to the registry, you must use the registry API functions, or for Visual Basic 4 and 5 users, there is a function set called Get- and SaveSettings which save to a specific portion of the registry: (HKEY_CURRENT_USER\Software\VB and VBA Program Settings) If you go the API route you’ll have a little work ahead of you because while the registry is stored in a tree-style format, you access each individual key, sub-key or value, in a fashion which resembles somewhat regular file access. (i.e., open, read and/or write, close.)

I recommend creating these keys and values in the registry with the registry editor (regedt32.exe) before trying to write these yourself if you are unfamiliar or uncomfortable with editing the registry. If you use GetSetting and/or SaveSetting, there is no way to hurt anything important in the registry.

Registry Access using Get- and SaveSetting

```
‘ Write an entry to:
‘ HKEY_CURRENT_USER\Software\VB and VBA Program Settings\MyApp\Settings\
SaveSetting "MyApp", "Settings", "Left", 10

‘ Read an entry and output to the Debug window.
Debug.Print GetSetting("MyApp", "Settings", "Left", 4)
```

Registry Access using the Windows API

```
Option Explicit

Private Declare Function RegOpenKey Lib "advapi32.dll" Alias _
"RegOpenKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, _
phkResult As Long) As Long
Private Declare Function RegQueryValueExStr Lib "advapi32.dll" _
Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName _
As String, ByVal lpReserved As Long, lpType As Long, ByVal lpData As _
String, lpcbData As Long) As Long
Private Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hKey As _
Long) As Long
Private Const HKEY_CURRENT_USER = &H80000001
Private Const REG_SZ = 1&

Private Sub Command1_Click()
        Dim hKey As Long
        Dim lRet As Long, lenData As Long
        Dim sSubKey As String, sData As String

        sSubKey = “Control Panel\Colors\”

        ‘ Open the specified Key (just a branch at this point.)
        lRet = RegOpenKey(HKEY_CURRENT_USER, sSubKey, hKey)

        ‘ Query a Value in that open key.
        lenData = 255
        sData = Space$(lenData)
        lRet = RegQueryValueExStr(hKey, “ButtonFace”, 0, REG_SZ, _
                sData, lenData)
        ‘ Trim off the excess & display the returned value
        sData = Left$(lenData)
        Debug.Print “The value is: ” & sData

        ‘ Close the opened key.
        lRet = RegCloseKey(hKey)
End Sub
```

- Robert Houghtaling

3. So which is best to use?

There are no hard and fast rules to use when choosing a method for accessing the registry. Microsoft, in choosing to move focus to the registry, is naturally tending lean away from using INI files. INI files can be very convenient though, because they can be accessed much quicker than the registry and with slightly less overhead and work. For small applications, using an INI file (as opposed to the registry) might be a better solution; for example, INI files can be edited with any text editor.

If you plan on widely distributing your application, and you wish to keep multiple versions of your software on one system, with a little planning, using the registry to store user data is very convenient because branches (keys) in the registry are completely separate from each other. If you move from version 4.0 to 5.0 and have an entirely new set of options, those options won't interfere with the old options, because you'll be looking in separate areas of the registry.

```
HKEY_CURRENT_USER
    Software
        MyCompany
            MyProduct
                4.0
                    SomeOptions
                5.0
                    SomeOtherOptions
```

- Robert Houghtaling

There has become quite a controversy, in the programming community, over whether or not programmers should use the system registry in their programs. It has turned into a love/hate relationship, either you love the registry or you hate it. In all honesty the System Registry does have it's use, but most programmers that do use it, overuse it.

The largest controversy over the registry is how much ram it takes up. Currently i have about 2 megs of my ram being eaten up with the configurations settings of programs that aren't loaded. Why should the configuration settings for programs that are not loaded or haven't been loaded for quite some time be sitting in your ram taking up memory? It is not at all acceptable to waste memory in this manor.
I find it quite appalling how many companies push off their responsibilities to create the best possible programs they can, and instead go for the quick/easy solution, their programs perform poorly and they just tell the end-users it's their problem to deal with. Telling them and should buy more ram or a faster computer. Programmers should make better use of the ram, processors, and resources the users have rather then wasting it.

The system registry is a great tool for Multi-User environments, but using it for single user environments or using it to store lots of options is an overkill. Remember the information your program stores in the registry is loaded into the computer's ram whether or not your program is loaded. The less ram there is free the slower their system and programs will perform, because the more memory swapping and other such tasks the system has to do.

In reality the INI/CFG files are far from dead, more and more programmers are returning to them every day. It is faster, although only slightly depending on the system, to load all of your configuration settings from an ini file then to deal with the declares to and communications with system registry. Plus you save your users a lot of problems on their systems which they will appreciate.

General Rules for Registry Use:

For multi-user environments rather then storing all of the configuration settings in the system registry under their name, store one setting which points to the path of an ini file that contains all of the config settings for that particular user.

Store program passwords encrypted into the registry. It provides a little more security because most users are quite unfamiliar with it.

If you have globally shared options such as the program's location, installation date, or general data use the registry only if there are a couple of them you wish to save. If there are numerous settings, save your users the memory and store them in an ini file and just point to the ini file that contains all of the global options in the registry.

In general save your users the memory and put as much of your configuration settings in ini files as possible so that they are only loaded into memory when needed. 5kb may not seem like much but it adds up fast. As a programmer you have to keep in mind your program isn't the only program being run on the system and thus have to respect the programmers of the other applications and respect the user of the computer by not having your program horde memory that it doesn't need. If all programmers respected each other and respected the users by reducing memory required by their apps, then all of the applications on the system would perform better.

I have even seen some programs that load over 200+ kb of options into the registry that's just quite absurd. If you are making an FTP Client or other program that has lots of usernames and passwords store it in a config, data, or password file anything but the registry. There is absolutely no reason such memory should be used up by data for a program that isn't loaded at the time.

- Dave Sherman

D. Working With Databases

1. Why shouldn't I use the Data Control? What should I use then?

According to the VB4.0 help, "you can perform most data access operations using the Data control without writing any code at all. Data-aware controls bound to a Data control automatically display data from one or more fields for the current record or, in some cases, for a set of records on either side of the current record." This is the primary reason many people choose to use the Data control; the learning-curve for this control is very small.

However, the Data control also hides much of the functionality and flexibility of DAO from the programmer. The Data control adds a bit of overhead to your project because you are stuck with the things that the controls designers thought you'd need, which makes the Data control quite a bit slower than the equivalent DAO methods and objects.

If you feel you want the extended functionality and if speed is a concern, then using the DAO methods would be a better choice than the Data control. If you need ease of use and a slightly faster development time (using DAO, after all is going to be a bit harder because you have to write the code yourself) then the Data control will be a better choice.

- Robert Houghtaling

2. Why is there only one record in my recordset?

If you did a recordcount and it says only one record is in your recordset (When you know there are more.)

let's say you had:

```
Dim rs As Recordset
```

then simply do a:

```
rs.MoveLast
```

and now your RecordCount will be correct. Don't forget to do a rs.MoveFirst to get back to the first record again.
- Guy Thompson

3. How do I put a picture in my database?

   I am unable to find a programmatic method of storing a picture in a jet database using visual basic. The only way I know is to open the database in access, and on a valid OleObject field, open the Insert Object dialog (Edit/Insert Object) and pick the file to insert.
   - Charles Haeberle

4. How do I add an ODBC data source in code?

   The Jet Engine RegisterDatabase method can be used to define new ODBC data sources. For more details, see the Visual Basic help files entry for 'RegisterDatabase Method'.
   - Charles Haeberle

E. Working With Graphics

1. How do I load/display a picture off my hard drive?

   A picture may be loaded and displayed by using a picture box control. Simply add a picture box to your project at design time, set the picture property of the picture box to the filename of the bitmap image you wish to display. Visual Basic 5 now supports JPG and GIF, so you can also use those file types if you are using a VB5 picture box.
   - Charles Haeberle

   At run time in order to load a picture you must use the LoadPicture Function. You can load a picture into a form, image control, picturebox, or variable. Below is an example of loading a picture named MyPicture.GIF into a picturebox.

```
        Picture1.Picture = LoadPicture("C:\Windows\MyPicture.BMP")
```

   - Dave Sherman

2. How can I put a picture on a button?

   Visual Basic 5 command buttons now have both a Picture, DownPicture and DisabledPicture property. Picture identifies the image to be displayed normally (including JPEG and GIFs), and if DownPicture is set, it will be displayed when the button is pressed. DisabledPicture, if set, identifies the graphic to display when the button is not enabled. The Style property of the command button must be set to 1 (Graphical) for these properties to do anything.
   - Charles Haeberle

3. How do I put bitmaps on menus?

   Declare the following Windows API functions and Constants in a module:

```
Public Const MF_BITMAP = &H4
Public Const CLR_MENUBAR = &H80000004

Const Number_of_Menu_Selections = 3
'changes depending on the number of menu items

Declare Function GetMenu Lib "user32" Alias "GetMenu" (ByVal hwnd As Long) _
As Long
Declare Function GetMenuItemID Lib "user32" Alias "GetMenuItemID" (ByVal _
 hMenu As Long, ByVal nPos As Long) As Long
Declare Function ModifyMenu Lib "user32" Alias "ModifyMenuA" (ByVal hMenu _
As Long, ByVal nPosition As Long, ByVal wFlags As Long, ByVal wIDNewItem _
As Long, ByVal lpString As String) As Long
Declare Function SetMenuItemBitmaps Lib "user32" Alias "SetMenuItemBitmaps" _
(ByVal hMenu As Long, ByVal nPosition As Long, ByVal wFlags As Long, ByVal _
hBitmapUnchecked As Long, ByVal hBitmapChecked As Long) As Long
```

The following code shows how to use both static and dynamic bitmaps for the menu.

Design a menu system using the Menu Design window such as the following:

Caption	Control Name	Indented	Index
BitMenu	TopMenu	No	
Sub Menu0	SubMenu	Once	0
Sub Menu1	SubMenu	Once	1
Sub Menu2	SubMenu	Once	2

Now create a picture control array with three bitmaps by creating three picture controls with the same control Name using the Properties list box.

Control Name	Caption	Index	FontSize
Picture1		0	N/A
Picture1		1	N/A
Picture1		2	N/A
Picture2		N/A	N/A 'check BMP
Picture3		0	'set Picture3 FontSize all different
Picture3		1	9.75
Picture3		2	18
Command1	Static		
Command2	Dynamic		

For each control index of Picture1, add a valid bitmap to the Picture property. Because these bitmaps will be displayed in the menu, you should use smaller bitmaps. Add a bitmap to the Picture2 Picture property that you want to be your check mark when you select a menu option.

And finally add the following code to your project:

```
Sub SubMenu_Click (Index As Integer)
        ' Uncheck presently checked item, check new item
        Static LastSelection As Integer
        SubMenu(LastSelection).Checked = False
        SubMenu(Index).Checked = True
        LastSelection = Index
End Sub

Sub Command1_Click ()
        Dim i As Integer
        Dim x As Long
        Dim hMenu As Long
        Dim hSubMenu As Long
        Dim menuId As Long

        'to create a static bitmap menu
        hMenu = GetMenu(Me.hWnd)
        hSubMenu = GetSubMenu(hMenu, 0)
        For i = 0 To Number_of_Menu_Selections - 1
                menuId = GetMenuItemID(hSubMenu, i)
                x = ModifyMenu(hMenu, menuId, MF_BITMAP, menuId, _
                CLng(picture1(i).Picture))
                x = SetMenuItemBitmaps(hMenu, menuId, 0, 0, _
                CLng(picture2.Picture))
        Next
End Sub
```

```
Sub Command2_Click ()
        Dim i As Integer
        Dim x As Long
        Dim hMenu As Long
        Dim hSubMenu As Long
        Dim menuId As Long

        'to create a dynamic menu system
        hMenu = GetMenu(Me.hWnd)
        hSubMenu = GetSubMenu(hMenu, 0)
        For i = 0 To Number_of_Menu_Selections - 1
                'Place some text into the menu.
                SubMenu(i).Caption = Picture3(i).FontName & _
                Str$(Picture3(i).FontSize) + " Pnt"

                '1. Must be AutoRedraw for Image().
                '2. Set Backcolor of Picture control to that of the
                ' current system Menu Bar color, so Dynamic bitmaps will appear
                ‘ as normal menu items when menu bar color is changed via the
                ‘ control panel
                '3. See the bitmaps on screen, this could all be done at design time.

                Picture3(i).AutoRedraw = True
                Picture3(i).BackColor = CLR_MENUBAR

                ' You can uncomment this
                ' Picture3(i).Visible = False

                ' Set the width and height of the Picture controls based on their
                ‘ corresponding Menu items caption, and the Picture controls
                ‘ Font and FontSize.
                'DoEvents() is necessary to make new dimension
                'values to take affect prior to exiting this Sub.

                Picture3(i).Width = Picture3(i).TextWidth(SubMenu(i).Caption)
                Picture3(i).Height = Picture3(i).TextHeight(SubMenu(i).Caption)
                Picture3(i).Print SubMenu(i).Caption

                'Set picture controls backgroup picture (Bitmap) to its Image.
                Picture3(i).Picture = Picture3(i).Image
                x = DoEvents()
        Next

        'Get handle to forms menu.
        hMenu = GetMenu(Me.hWnd)

        'Get handle to the specific menu in top level menu.
        hSubMenu = GetSubMenu(hMenu, 0)
        For i = 0 To Number_of_Menu_Selections - 1
                'Get ID of sub menu
                menuId = GetMenuItemID(hSubMenu, i)

                'Replace menu text w/bitmap from corresponding picture control
                x = ModifyMenu(hMenu, menuId, MF_BITMAP, menuId, _
                CLng(Picture3(i).Picture))
                'Replace bitmap for menu check mark with custom check bitmap
                x = SetMenuItemBitmaps(hMenu, menuId, 0, 0, _
                CLng(picture2.Picture))
        Next
End Sub
```

- Guy Thompson

4. How can I tile a bitmap?

There are two ways to do this, one being internally in vb and the other being with the Windows API. VB4's internal structures to do this by far out benchmarked the API equivalent. I haven't had time to benchmark it in VB5 but odds are with VB5 it's even faster then it was in VB4.

In this example create a new project. On form1 place a picturebox. Set the picturebox's picture property to a picture file, for now keep it simple just refer it to like WINDOWS\TILES.BMP later you can change it to whatever.

```
    Private Sub Form_Load()

        ' Prepare the form and picturebox
        Form1.ScaleMode = 3
        Form1.AutoRedraw = True
        Picture1.ScaleMode = 3

        ' Get dimensions
        Dim FormHeight As Long
        Dim FormWidth As Long
        Dim PictureHeight As Long
        Dim PictureWidth As Long

        ' Assign dimensions
        FormHeight = Form1.ScaleHeight
        FormWidth = Form1.ScaleWidth
        PictureHeight = Picture1.ScaleHeight
        PictureWidth = Picture1.ScaleWidth

        ' Tile bitmap
        For y = 0 To FormHeight Step PictureHeight
            For x = 0 To FormWidth Step PictureWidth
                Form1.PaintPicture Picture1.Picture, x, y
            Next x
        Next y
    End Sub
```

- Dave Sherman

5. How can I tile a bitmap on a MDI form?

In this example create a new project. Create a MDI form and set it to be your startup form. Place a picturebox so it docks into the top part of the mdiform. Now set the mdiform’s picture property to a picture file, for instance WINDOWS\TILES.BMP.

```
Private Sub MDIForm_Load()
        ' Prepare form
        Picture1.AutoRedraw = True
        Picture1.Visible = False

        ' Get dimensions
        Dim FormHeight As Long
        Dim FormWidth As Long
        Dim PictureHeight As Long
        Dim PictureWidth As Long

        ' Assign dimensions
        FormHeight = MDIForm1.Height
        FormWidth = MDIForm1.Width
        PictureHeight = Picture1.ScaleX(MDIForm1.Picture.Height, 8, 1)
        PictureWidth = Picture1.ScaleY(MDIForm1.Picture.Width, 8, 1)

        'Resize picturebox
        Picture1.Height = MDIForm1.Height

        ' Create a new tiled form of the bitmap
        For y = 0 To FormHeight Step PictureHeight
                For x = 0 To FormWidth Step PictureWidth
                        Picture1.PaintPicture MDIForm1.Picture, x, y
                Next x
        Next y

        ‘ Copy our new bitmap to the back of the MDIFrom
        MDIForm1.Picture = Picture1.Image
End Sub
```

- Dave Sherman

F. Working With Multimedia

1. How do I play an animation?

Playing a video on your form isn't as difficult as it looks. Place one PictureBox on your form and size it to approximately the size of the animation you wish to play. Also put a CommandButton and a Multimedia Control on your form. I added the following code to a form: (choose an AVI file from your HDD)

```
Private Sub Command1_Click()
        With MMControl1
                .hWndDisplay = Picture1.hWnd
                .filename = "c:\movies\myvideo.avi"
                .DeviceType = "AVIVideo"
                .Command = "open"
                .Command = "play"
        End With
End Sub
```

When you click the command button, the video (with sound, if there is any) should play in the PictureBox. If the picturebox is too small or too large, resize it to your needs at design-time.
- Robert Houghtaling

2. How do I play Midis?

There is an API function, documented by MS on their knowledge base, called mciSendString which allows you to play MIDI files synchronously and asynchronously.

Unfortunately, VB does not allow callbacks without an external OLE control, so there is no way for the multimedia subsystem to notify you when a particular .MID file is done playing. This means that you can play a MIDIfile once and pause until it finishes or you can start the file playing and continue immediately, but never actually know when it finishes.

```
Private Declare Function mciSendString Lib "winmm.dll" Alias _
"mciSendStringA" (ByVal lpstrCommand As String, _
ByVal lpstrReturnString As String, ByVal uReturnLength As Long, _
ByVal hwndCallback As Long) As Long

Dim ret As Long

ret = mciSendString("open c:\midi\myfile.mid type sequencer alias myfile", 0&, 0, 0)
ret = mciSendString("play myfile wait", 0&, 0, 0)
ret = mciSendString("close myfile", 0&, 0, 0)
```

An alternative is to use the Microsoft Multimedia Control. With it you can play AVI files, MIDI files, Wave files, etc.

In the example below, I added one Multimedia Control and one command button to a form. Then I added the following code (choose a MIDI file from your HDD) …when the button is clicked, the MIDI file will start to play and will repeat until the program is closed. It does this by using the MMControl's _Done event.

```
Private Sub Command1_Click()
        With MMControl1
                .filename = "c:\midi\myfile.mid"
                .DeviceType = "sequencer"
                .Command = "open"
                .Command = "play"
        End With
End Sub
```

```
Private Sub MMControl1_Done(NotifyCode As Integer)
        If NotifyCode = 1 Then ' Successful
                MMControl1.Command = "play"
        End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
        MMControl1.Command = "Close"
End Sub
```

- Robert Houghtaling

3. How do I play Waves?

The easiest way I have found to play .WAV files is to use the API function sndPlaySound().

```
Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" _
(ByVal lpszSoundName As String, ByVal uFlags As Long) As Long

Private Const SND_ASYNC = &H1

sndPlaySound "c:\windows\chime.wav", SND_ASYNC
```

There are many other options to playing wave files, including looping, synchronous playing, asynchronous playing, playing from memory and playing system wave sounds like SystemExit.

If you would like to play a wave file using the Microsoft Multimedia Control, drop one a form along with a command button and add this code (choose a wave file from your HDD):

```
Private Sub Command1_Click()
        With MMControl1
                .filename = "c:\waves\mywave.wav"
                .DeviceType = "waveaudio"
                .Command = "open"
                .Command = "play"
        End With
End Sub
```

- Robert Houghtaling

G. Miscellaneous

1. How do I generate random numbers?

From the Visual Basic 5.0 Help file:

The Rnd function returns a value less than 1 but greater than or equal to zero. The value of number determines how Rnd generates a random number:For any given initial seed, the same number sequence is generated because each successive call to the Rnd function uses the previous number as a seed for the next number in the sequence. Before calling Rnd, use the Randomize statement without an argument to initialize the random-number generator with a seed based on the system timer.

To produce random integers in a given range, use this formula:

```
Int((upperbound - lowerbound + 1) * Rnd + lowerbound)
```

Here, upperbound is the highest number in the range, and lowerbound is the lowest number in the range.

Note   To repeat sequences of random numbers, call Rnd with a negative argument immediately before using Randomize with a numeric argument. Using Randomize with the same value for number does not repeat the previous sequence.
- Charles Haeberle

2. How do I center a form on the display?

Visual Basic 5 now provides a StartUpPosition property for all forms. By setting the StartUpPosition property, you can have the form automatically be centered. StartUpPosition may be one of four values:

vbStartUpManual	0	No initial setting specified.
vbStartUpOwner	1	Center on the item to which the UserForm belongs.
vbStartUpScreen	2	Center on the whole screen.
vbStartUpWindowsDefault	3	Position in upper-left corner of screen.

- Charles Haeberle

3. How do I pause a program?

If you want to pause for a certain amount of time, the best way is to use the Sleep() API function.

```
Declare Sub Sleep Lib "kernel32" Alias "Sleep" (ByVal dwMilliseconds As Long)

Call Sleep(5000)
```

This will pause your program for 5 seconds, while still allowing other windows events to continue.
- Robert Houghtaling

Personally I don’t much care for the Sleep() API Function because it completely freezes your application, not allowing for any part of the program to function while paused. What this means? Well it means that your program won’t be updated visually.. if another program covers it and then uncovers it you’ll see pretty much nothing but a white box if that because the program can’t refresh it’s screen or the way it looks. What else this means is the user can’t click or do anything to get the program to unpause or to perform some other task. Also if you have any timers or any routines that are supposed to function on a certain timed basis like a program event of some sort it can also not execute.

The way I get around this is using my own pause techniques. In Visual Basic, when you call one function while another is in execution they new one will take precedence and will execute not returning control to the original one until the new function is done. In the example below, I made compensation for if it goes over the midnight mark once, and I used an integer for the pause seconds because there is no reason why you should ever need more then a 32,767 second pause during a procedure's execution.

```
Public Sub PauseProg(PauseTime As Long)
        StartTime = Timer
        If StartTime + PauseTime > 86399 Then
                PauseTime = StartTime + PauseTime - 86400
                Do While Timer <= 36399
                        DoEvents
                Loop
                Do While Timer < PauseTime
                        DoEvents
                Loop
        Else
                Do While Timer < StartTime + PauseTime
                        DoEvents
                Loop
        End If
End Sub
```

Just as a note many programs that report the processor in use will report back saying that 100% processor is in use when using this pause technique. Our program is the program being credited with all of that processing time, this is rather misleading however because well over 90% of the processor usage in our routine is being pumped right back into the os for it to do with as it wants and for it to handle it's events. So don't pay that much attention it's the monitors that are detecting processor usage can't compensate for that.

You can now pause your program by doing a call to this routine.

```
Call PauseProg(20) ' Would pause the program for 20 seconds
```

A small word of warning, you have to be careful when using this technique because it uses the DoEvents Function. What I mean by this is there are some events you won't want executed while pausing in this method. For example if you execute this in a click event of a button you don't want the same button pushed during this time. A couple ways to get around this is to make a global variable that you set in the pause routine and check for at the beginning of the click event routine. If the variable has a value you would have the click event automatically exit the sub returning control back to the pause routine. If this sounds to complicated just set the buttons you don't want the user to click on, while this is executing, to false.
- Dave Sherman

4. How do I run a program from VB?

To launch another application from Visual Basic you use the Shell statement.

```
Shell "program name", 1
```

The one problem with the Shell statement is that unless the file is in the path, you need to include it with the call to shell.

An alternative is the ShellExecute API function. With ShellExecute, you can launch everything from programs to text files to internet web pages.

```
Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _
(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, _
ByVal lpParameters As String, ByVal lpDirectory As String, _
ByVal nShowCmd As Long) As Long

' Launch notepad.exe
ShellExecute Me.hWnd, "Open", "notepad.exe", "", "", 1
```

```
' Launch win.ini with the default editor
ShellExecute Me.hWnd, "Edit", "win.ini", "", "", 1

' Launch www.microsoft.com with the default WWW browser.
'   Notice you do not need the http:// qualifier.
ShellExecute Me.hWnd, "Open", "www.microsoft.com", "", "", 1
```

- Robert Houghtaling

5. How do I pause until it's done?

Now here is where it starts to get complicated, so pay close attention. There are, by necessity, two different methods to pause a program until a shelled process completes; one for 16-bit Windows and one for 32-bit Windows.

There are two comprehensive Microsoft Knowledge Base articles that deal with this task.

16-Bit Article (this article will need a little adjustment to work as a pause routine. It uses the VB Shell statement to launch an app and then immediately displays the window caption of the shelled app. You simply loop at that point, testing for the existence of the shelled app's window. All the code necessary to do so is included with the KB article)

How to Find a Window Handle Based on an Instance Handle
http://www.microsoft.com/kb/articles/q127/0/30.htm

32-Bit Article

How a 32-Bit App Can Determine When a Shelled Process Ends
http://www.microsoft.com/kb/articles/Q129/7/96.htm

- Robert Houghtaling

6. How do I shut it down now that it's done?

To preemptively shut down a shelled app in 32-bit Windows, you should use the CreateProcess() API function documented in the 32-bit Knowledge Base article above to shell to the application and then use the TerminateProcess() API function to shut that process down.

```
Dim rVal As Long
rVal = TerminateProcess(hProcess, 0)
```

hProcess is a Long value representing a handle to a process that was returned from the CreateProcess() API function.

Note: at this time, I have no firm documentation on shutting down external 16-bit applications (other than the remote possibility of using Visual Basic's SendKeys statement to send the Alt-F4 keystroke combination…which is chancy at best)

A side issue here is how to shell to a DOS application and have it shut down the Finished DOS window when the application has finished. To do this, specify "/C" on the command line to which you are shelling:

```
h = Shell("COMMAND.COM /C <pathname>")
```

By default, Windows95 leaves finished DOS windows open, waiting for you to close them. This will bypass that procedure and close the DOS window for you.

- Robert Houghtaling

7. How can I use a variable on more then one form?

   In order to create a variable so it can be accessed and modified by all of your forms, you must first add a module to your project. In the declaration section of the module, VB4 & VB5, you must declare the variable as a “Public” variable. Below are a few different global declare examples.

```
Public MyStr As String
Public MyInt As Integer
Public Temp As Variant
```

   - Dave Sherman

8. Is there any way to pass a variable to another form apart from using global variables?

   On the target form, place a text box, label, or any other control that you prefer and set that control's visible property to False. Then place the value of your variable in the text (or caption, etc.) property of this control. Another method is to use the .Tag property which is never used by VB itself. You can place any string you want into the .Tag property of a form or control.
   - Guy Thompson

9. Why does my program not close correctly when the user closes it? (It still shows up when I press Ctrl-alt-del.)

   This occurs when some part of the program is still running or loaded. For instance if you have two forms, you hide the first and close the second. While you see nothing the first form is still there just invisible. Same goes for modules, modules operate without a visible form so just because you see nothing doesn’t mean it’s not loaded or processing something.

   The way to correct this is to use the end statement. The end statement will terminate the entire program and unload it from memory. On a quit button or in your main forms unload event place the statement

```
End
```

   That one little word will completely terminate your program and unload all of it’s components.

   - Dave Sherman

10. How do I change the size of an array?

   Create a new project and on Form1 place a command button. Now add the below code to the form.

```
Private Sub Command1_Click()
        Dim FooBar() As String ‘ Declare our array open ended
        ReDim FooBar(1 To 15) ‘ Give our array boundries
        For x = LBound(FooBar) To UBound(FooBar)
                FooBar(x) = "I Am #" & x
                ‘ Loop through and assign each item in our array a value
        Next x
        ReDim FooBar(1 To 16) ‘ Redimension our array’s boundries
        FooBar(16) = "I Am #16" ‘ Give our new item a value
        For x = LBound(FooBar) To UBound(FooBar)
                Debug.Print x & ": " & FooBar(x)
                ‘ Loop through and print all the items to the debug box
        Next x
End Sub
```

When you click on the button you see a bunch of blank spaces being listed for items 1-15 and then our actual value we assigned listed for 16. This is because when we redimensioned our array we didn't' preserve all the existing data in it, so all the data was lost. Since we added the data for 16 after we changed the array's size it was never erased. See the following question if you wish to keep all the data in your array when you resize it.

- Dave Sherman

11. How do I change the size and not lose my data?

When you change the size of an array, you normally lose all of the data in it. However, VB has a keyword, called Preserve, that you can add to your ReDim statement so that you may keep all your data and only resize the actual array. Create a new project and on Form1 place a command button, then insert the code below.

```
Private Sub Command1_Click()
        Dim FooBar() As String ‘ Declare our array open ended
        ReDim FooBar(1 To 15) ‘ Give our array boundries
        For x = LBound(FooBar) To UBound(FooBar)
                FooBar(x) = "I Am #" & x
                ‘ Loop through and assign each item in our array a value
        Next x
        ReDim Preserve FooBar(1 To 16)
        ‘ Redimension our array's boundries preserving the data contained within
        FooBar(16) = "I Am #16" ‘ Give our new item a value
        For x = LBound(FooBar) To UBound(FooBar)
                Debug.Print x & ": " & FooBar(x)
                ‘ Loop through and print all the items to the debug box
        Next x
End Sub
```

When you run it this time the output lists all the items with their data still intact. So the key to remember here is when you want to keep the data in all your items make sure you use the Preserve keyword in the ReDim statement.

- Dave Sherman

12. How to find the path of the executable?

To find the path of the application use the path property of the app.

```
Debug.Print App.Path
```

- Dave Sherman

13. How do I call a help file from my program?

Most professional programs have an on-line help option on one of their pull down menus. There are a few ways to do this from using the common controls, to the API, to the simple shell command. Personally i prefer the shell command because it's the fastest of the 3 doesn't require any extra dlls/ocxes or ram allocation for the call.

```
Private Sub HelpMe_Click()
        RetVal = Shell("winhelp.exe " & App.Path & "\HELPFILE.HLP", 3)
End Sub
```

- Dave Sherman

14. Why does my program say wrong version of runtime DLL?

We have only seen this problem with VB 4. With Visual Basic 4.0 the runtime dll names were VB40032.DLL and VB40016.DLL. Microsoft then released an update version that many developers upgraded to, Visual Basic 4.0a. The problem occured when Microsoft kept the dll names the same. They strived very hard to keep the DLL backwards compatible, in terms of the VB4.0 programs could use the VB4.0a DLL without a problem. What they didn't take into consideration was that programs when installed would overwrite the 4.0a version with the 4.0 version since they were the same name. Now while 4.0 programs could use the 4.0a dll it was not the other way around, 4.0a programs couldn't use the 4.0 dll because the structures were slightly changed. So if you get this error message the solution is to replace the VB400xx.DLL in your WINDOWS/SYSTEM dir with the 4.0a version.

- Dave Sherman

15. Why can't my friend run the program I wrote?

One of major disadvantages of using a bunch of external custom controls or references in your program, is that your program will need all of them when you distribute it. Many new programmers overlook that all dlls and ocxes you use in your program might not be on the end user's computer. This is why when you distribute your application you should probably include all of the dlls and ocxes with it just incase the user don't have them.

Merley including some of them isn't good enough in some cases though. Some ocxes and dlls require registration in the system registry before they can be used. There are a couple ways to register them but the easiest way is to create a setup program that will install and register the dlls and ocxes.

Visual Basic comes with a program called Setup Wizard that creates custom installations. While it does work it has a very bad habit of including MANY dlls that just aren't needed. For those that need a better solution then what Setup Wizard provides you may with to make your own installation program or use a commercial program such as Wise, or Install Shield.

- Dave Sherman

16. How do I copy/retrieve things to/from the clipboard?

Using the clipboard is very simple. There are four basic methods you will use. For working with text, those methods are SetText and GetText, and for binary data, SetData and GetData. SetText and GetText are the easiest of the two. The following example copies a simple string to the clipboard, then puts that clipboard text into a text box.

```
Clipboard.SetText “Hello World”
MyTextBox.Text = Clipboard.GetText
```

Working with binary data is a bit more complicated, because the clipboard can support different data formats. Consult the Visual Basic Help files on the two topics, SetData and GetData, for an overview of use of the clipboard for working with binary data.
- Charles Haeberle

17. When I use the IIF function, I am getting errors in my program when I give the program to a friend...why?

Microsoft placed financial functions in a separate DLL called, MSAFINX.DLL. If you include that dll with your source you shouldn't have any problems. Why is IIF considered a financial function? Who knows?...if you find an answer tell me...

Personally, I do not use IIF because it is slow (extremely, when compared to the equivalent IF statement) and I don't really want to give out another DLL.
- Robert Houghtaling

V. Advanced Questions/Issues With Visual Basic

A. General Questions

1. How do I make my window always stay on top?

   Making a form, or window, stay above all other windows requires a little API calling. The function needed is the **SetWindowPos** function. To make this function available for use in your program, you must have a public module in your application. In the General Declarations section of the public module, add these lines:

```
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal _
hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, ByVal cx _
As Long, ByVal cy As Long, ByVal wFlags As Long) As Long

Public Const HWND_TOPMOST = -1
Public Const SWP_NOMOVE = &H2
Public Const SWP_NOSIZE = &H1
Public Const SWP_NOZORDER = &H8
```

   Next, add the following public subroutine to your module:

```
Public Sub SetFormTopmost(TheForm As Form)
        SetWindowPos TheForm.hwnd, HWND_TOPMOST, 0, 0, 0, 0, _
        SWP_NOZORDER + SWP_NOMOVE + SWP_NOSIZE
End Sub
```

   Finally, set the form to topmost by calling the function SetFormTopmost with the name of the form to be set topmost as the parameter. For example, if frmMyForm is the form to be set to topmost:

```
SetFormTopmost frmMyForm.
```

   By calling the SetWindowPos function, we send windows a message saying – take this window, put it on top, ignore the size and position parameters, and don't allow it to lose its topmost position. It should be noted that in the API viewer which comes with Visual Basic incorrectly defines the value of SWP_NOZORDER as &H4. SWP_NOZORDER is &H8, and must be set as such in your project for this function to work. For more information on the SetWindowPos function and its other uses, and you have internet access, take a look at:
   http://www.microsoft.com/msdn/sdk/platforms/doc/sdk/win32/func/src/f84_2.htm.
   - Charles Haeberle

2. How can I set a minimum size a user can resize my form to?

   In the forms Resize event, if the width or height has gone below or above the desired ranges, they can be set there. For example, to prevent a form from being made shorter than 1000 units or taller than 10000:

```
Private Sub MyForm_Resize()
        Select Case Me.Height
        Case Is < 1000
                Me.Height = 1000
        Case Is > 10000
                Me.Height = 10000
        End Select
End Sub
```

   - Charles Haeberle

3. How do I shutdown/restart windows?

16-bit Windows uses the ExitWindows() API function and 32-bit Windows uses ExitWindowsEx(). The 32-bit version has many more options than its 16-bit equivalent including the ability to logoff and shutdown.

***** 16-bit Sample *****
In a project with 1 commandbutton, place the following code:

```
Declare Function ExitWindows Lib "user" (ByVal uFlags As Long, ByVal _
dwReserved As integer) As integer
Const EW_REBOOTSYSTEM = &H43
Const EW_RESTARTWINDOWS = &H42

Sub Command1_Click()
        Dim iAns As Integer
        Dim rVal As Integer
        Dim iButtonType as integer

        iButtonType = 4 + 32 ' vbYesNo + vbQuestion

        ' Ask if the user is sure they want to exit.
        iAns = MsgBox("Are you sure you want to exit windows?", iButtonType, _
        "Exit Windows")

        If iAns = 6 Then ' Yes pressed
                ' Call the exit function to Reboot.
                rVal = ExitWindows(EW_REBOOTSYSTEM, 0)
        End If
End Sub
```

***** 32-bit Sample *****
In a project with 1 commandbutton, place the following code:

```
Private Declare Function ExitWindowsEx Lib "user32" (ByVal uFlags As Long, _
ByVal dwReserved As Long) As Long
Private Const EWX_LOGOFF = 0
Private Const EWX_SHUTDOWN = 1
Private Const EWX_REBOOT = 2
Private Const EWX_FORCE = 4

Private Sub Command1_Click()
        Dim iAns As Integer
        Dim rVal As Long

        ' Ask if the user is sure they want to exit.
        iAns = MsgBox("Are you sure you want to exit windows?", vbQuestion Or _
        vbYesNo, "Exit Windows")

        If iAns = vbYes Then ' Yes pressed
                ' Call the exit function to ShutDown.
                rVal = ExitWindowsEx(EWX_SHUTDOWN, 0&)
        End If
End Sub
```

- Robert Houghtaling

4. How do I create/delete/change a shortcut?

   Creating a shortcut (a Shell Link) is rather easy with Visual Basic, because Microsoft created a simple API function in the setupkit DLL called fCreateShellLink. Deleting a Shell Link is even easier, because all that need be done is delete the correct .LNK file. Alas, changing a shortcut is not directly available with Visual Basic because you must go through the Windows95 IShellLink interface to access the different parameters of a Shell Link.

   To create a shell link, add the following code to the default form of a new project:

```
Private Declare Function fCreateShellLink Lib "STKIT432.DLL" (ByVal _
lpstrFolderName As String, ByVal lpstrLinkName As String, ByVal _
lpstrLinkPath As String, ByVal lpstrLinkArgs As String) As Long

Sub Command1_Click()
        Dim lReturn As Long

        'Add to Desktop
        lReturn = fCreateShellLink("..\..\Desktop", _
        "Shortcut to Calculator", "c:\windows\calc.exe", "")
        'Add to Program Menu Group
        lReturn = fCreateShellLink("", "Shortcut to Calculator", _
        "c:\windows\calc.exe", "")
        'Add to Startup Group
        lReturn = fCreateShellLink("\Startup", "Shortcut to Calculator", _
        "c:\windows\calc.exe", "")
End Sub
```

   - Robert Houghtaling

5. How do I work with the dialup networking stuff?

   Dial-up networking is accessible through the Windows API and again our friend Microsoft has come up with a very nice demo program for all of us which uses a remote automation server and VB4.

   ftp://ftp.microsoft.com/softlib/mslfiles/vb32ras.exe

   This demo program is complete and very interesting. It not only shows how to access DUN, it is a good example of using an OLE Server in VB4.
   - Robert Houghtaling

6. How do I make help files?

   A Help file is really nothing more than an RTF-formatted file with special tags that let a help compiler turn it into a .hlp file.

   The first step is to find a program which will compile help files. Hey! What do you know? Visual Basic comes with one: HC.EXE

   Next you have to create the help source files...I recommend getting a commercial package as these generally make the best help files...they automate a lot of the grunt-work associated with help files like the help topics, indexing, popups, etc.

   If you cannot afford a commercial package, the next step is a shareware or freeware package. These will be somewhat difficult to find generally, but most search engines return hits to the more common ones.

   After you have designed and compiled your .HLP file, you have to add the tags to your Visual Basic program so that when the user presses F1 on your forms, something actually happens.

Most objects in VB have a .HelpContextID property which is just a Long Integer. Many help design packages will output a .BAS file for you to import into your project which contains a list of Public Constants that is all of the ContextIDs in the help file. You can use these constants in your code (for example a HelpInit subroutine called from Form_Load) to set these objects.

- Robert Houghtaling

7. How do you duplicate forms like mIRC does?

Duplicating forms seems more complicated then it really is. Create a new project, and on form1 place a command button. Next add the code below to the command button.

```
Private Sub Command1_Click()
        Dim Form2 As New Form1
        Form2.Show
End Sub
```

This shows a very basic method used to duplicate a form then show it. In a real world application you would probably wish to change the properties, such as the form's caption and position, of this new form before you show it.

-Dave Sherman

8. How do I make my program not close when someone pushes the x?

In Visual Basic there are two unload events for every form. They are executed whenever the form is about to be unloaded. The major difference is the QueryUnload event allows one to cancel the process and even tell the program what was done to cause the form to start being unloaded. To cancel the close/unload of a form put the following in the QueryUnload event of the form.

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
        Cancel = -1
End Sub
```

-Dave Sherman

9. How do I make the mouse cursor invisible?

There are two real methods one can use to accomplish this. One is by using the ShowCursor() API. Both accomplish this by the same method, they set the mouse cursor to an invisible cursor. The mouse cursor is still there and you can still click, but you can't see the cursor so the user won't know where the pointer is or know where to click.

```
Declare Function ShowCursor Lib "user32" Alias "ShowCursor" (ByVal bShow As _
Long) As Long ' Place this in the declare section of a module

RetFunct = ShowCursor(False) ' This is the call to hide the mouse cursor
RetFunct = ShowCursor(True) ' This is the call to show the mouse cursor again
```

The other method involves including an icon that is made up of nothing but invisible pixels, then set the MouseIcon to that picture, and then setting the MousePointer property to 99.

- Dave Sherman

10. Can a VB Application be an OLE server?

Yes, to create an OLE server:

a. Decide what objects you need to define and their relationship to one another. You also need to decide just what properties and methods each of your classes must have.
b. Define the individual classes in the OLE server.
c. In the Project tab of the Options dialog box, set the Project Name, StartMode, and Application Description options for the OLE server.
d. Prepare the initialization of the OLE server. If the OLE server doesn't display any forms and only provides access to the objects in code, set the Startup Form option in the Project tab of the Options dialog box to Sub Main. Any initializing code for the OLE server should go in a Sub Main procedure in a standard module.

- **Guy Thompson**

11. How do I remove my program from the tasklist/taskbar?

In a module declare:

```
Public Const SW_HIDE = 0
Public Const GW_OWNER = 4
Declare Function GetWindow Lib "user32" Alias "GetWindow" (ByVal hwnd As _
Long, ByVal wCmd As Long) As Long
Declare Function ShowWindow Lib "user32" Alias "ShowWindow" (ByVal hwnd _
As Long, ByVal nCmdShow As Long) As Long
```

Then in your Form_Load() place the following code:

```
Dim rc As Long
Dim OwnerhWnd As Long

'make the form invisible
Me.Visible = False

'now remove it from the Task Manager List
OwnerhWnd = GetWindow(Me.hWnd, GW_OWNER)
rc = ShowWindow(OwnerhWnd, SW_HIDE)
```

- **Guy Thompson**

B. Coding Techniques

1. Why should I use Option Explicit?

Option Explicit is used in the General Declarations section of any module (Class, Form, General, etc.) to direct the compiler not to allow hot variable declaration. Hot, or on-the-fly, variable declaration is the practice of inventing new variables within code without any formal declaration statements. While this can be very convenient, ultimately it can lead to many more troubles than those which are solved by requiring formal declarations.

First of all, there’s simple syntax checking. Nobody is a perfect typist. We all make mistakes. When you are programming, mistakes can lead to unpredictable results. If you are properly using self documenting variable names, like LoopIndex, ClassroomCount, or MaximumOribitCircumference, to name a few examples, then the risk of misspelling one of these variables becomes much greater. However, rather than use this as an argument for using simple variable names like x, y and z, it becomes an argument for Option Explicit. When variables **must** be declared, the Visual Basic compiler registers an error if you use an unrecognized variable name. So if you type Fiend when you meant Friend, instead of your program acting like the former, it will instead warn you at compile that the variable Fiend is undeclared. This might be a bit of brutal truth, but what are Friends for?

In addition to the hours of debugging that can be avoided by using Option Explicit, proper variable declaration helps to save resources, time, and encourages more efficient programming by forcing the programmer to plan out procedure operations in advance.

There's one other benefit to Option Explicit which should be mentioned here. If you don't use it, your life expectancy may be prematurely decreased in a significant fashion by whoever takes over code maintenance after you move on to your next project. :-)
- Charles Haeberle

2. What is passing by reference and by value?

When a variable is passed to a procedure, it may be passed in one of two ways: By Reference or By Value.

A variable is really a block of memory that contains a value, and has a name by which it may be referred to (the variable name). Passing a variable by value means that the number or string stored in memory is passed to the procedure, but the original variable in the calling procedure can not be affected. Visual Basic actually creates a new instance of that variable for use by the receiving procedure.

Passing a variable by reference, on the other hand, allows the procedure which receives the variable to actually change the variable's value in the calling function. Visual Basic passes the address in memory of the variable to the procedure, and changes made to that value are reflected in the function which called it.

Here's an example.

```
Private Sub NoChange(ByVal ThisVal as Variant)
        Debug.Print ThisVal ‘ Received literal 5
        ThisVal = ThisVal + 1
End Sub

Private Sub DoesChange(ByRef ThisVal as Variant)
        Debug.Print ThisVal ‘ Received address of ThisVal
        ThisVal = ThisVal + 1
End Sub

Sub Main()
        Dim x as Integer
        X = 5
        NoChange X
        Debug.Print X  ‘ X still = 5
        DoesChange X
        Debug.Print X ‘ X now = 6
End Sub
```

By default, Visual Basic passes all variables by Reference, not by value. However, it is generally not recommended that functions change the values they receive, because if the change to the parameter is not known by the calling function, there can be unpredictable results. Typical examples of functions which would **properly** change a parameter would be: StripSpaces(thestring), ChangeDblQuotetoSingleQuote(TheString), IncrementByOne(TheValue).
- Charles Haeberle

3. Why should/shouldn't I use gosub and goto?

Goto and Gosub are the wicked step children left over from the early days of BASIC when it was not nearly as structured or object-oriented as it is now. These statements violate normal, logical program flow (if you've ever seen a nice, complicated program written in BASIC, you'll know what the term "spaghetti code" really means.)

I have never needed to use Goto in Visual Basic; there are always alternatives. I seriously recommend staying away from Gosub and Goto except for one case. Error Trapping--trapping errors in Visual Basic is not something that follows structured programming rules.

When you error trap, you have the option whether to use Goto:

```
*** Code fragment ***

On Error Resume Next

' Do something which may cause an error
' Test for errors & react to them if necessary
' Continue processing

*** or ***

On Error Goto MyErrorTrap

' Do something which may cause an error
' If an error was raised, program flow jumps down to the error trap.

' Continue processing

Exit Sub

MyErrorTrap:
' decide what error was raised & do something about it.
Resume Next

*** End Code fragment ***
```

The differences in the two techniques are minimal...one conveniently negates the need for Goto; the other is more structured. It will ultimately be your decision to pick a method...whichever you are more comfortable with.

- Robert Houghtaling

C. Windows API

1. What is the Windows API?

The Windows API is, at its heart, a set of DLLs that contain functions and subroutines that are used in common throughout the system. Generally speaking, those are the User, GDI, Kernel and Shell services. There are many other minor services like OLE, DDEML, Multimedia, LZW (compression), ToolHelp and Registration, but these aren't used quite as often.

User functions deal with the management of windows, message dispatching, threads, string handling and all types of user input device management.

Kernel functions handle the more low-level functions such as: memory allocation (real, heap and virtual), process management, loading/unloading resources and disk and comm I/O control.

GDI functions are the workhorse functions which draw and manage the graphical elements of the user-interface. Fonts, palettes, device contexts, pens, brushes, bitmaps, metafiles, regions, polygons, windows and viewports are some of the many object types handled by the GDI.

The Shell API contains many useful routines that are of a much higher-level type than the other routines. They generally encapsulate an idea rather than a small, logical step in a process. For instance, the GDI function LineTo draws a line from one point to another, while the Shell function SHFileOperation, manipulates files and folders (copy, move, rename, etc.) including the animation, progress bar and cancel button, all from one short function call. This makes the Shell functions much more powerful, but less flexible. Other objects controlled by the Shell API are icons, shell links (shortcuts), environment handling and drag-and-drop control.
- Robert Houghtaling

2. How do I use the Windows API?

To call an API function in VB, you must first Declare it in your code. In VB5 the API function declarations will be vb\winapi\APILOAD.EXE. In VB4, the API function declarations can be found in your vb\winapi\ directory using a program called APILODxx.EXE (where the xx is 16 or 32). If you don't have this program because you have the Standard version of VB4, you can try searching the Microsoft Knowledge Base. In VB3, the API is shown in a help file called win31api.hlp, but no actual VB declarations are shown, just the original C/C++ equivalent.

NOTE: There is a lot of good information in the text file, "vb4dll.txt". VB3 has a DLL text file, too, but I don't know its name.

Below are two example API function declarations; GetCurrentDirectory() and Sleep().

```
Declare Function GetCurrentDirectory Lib "kernel32" (ByVal nBufferLength As Long, _
ByVal lpBuffer As String) As Long
Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

GetCurrentDirectory() will retrieve the current directory in the same way that the DOS 'CD' command will. The Sleep() subroutine will pause your program for the specified number of milliseconds (while still allowing other applications to run normally).

You place these declarations in the General Declarations section of either a form or a .BAS module. If you want the function to be available to the entire program, you would place it in the .BAS module and prefix the declare with either 'Public' or 'Global' (vb5/vb4 & vb3). If you want it only to be available to a specific form, you use 'Private' (vb5/vb4).

- .BAS module
(vb5/vb4) - Public Declare Function ...
(vb3) - Global Declare Function ...

- .FRM file
(vb5/vb4) - Private Declare Function ...
(vb3) - Declare Function ...

Calling API functions can be tricky if you don't understand how they work. Since they follow the C/C++ style of calling, you have to follow their rules; the rules you know for calling VB functions and subs won't always work.

For example, the Sleep() API function is just one call with one numeric parameter and no return value, so is very straightforward.

```
Call Sleep(5000) ' Pause for 5 seconds.
```

On the other hand, a function like GetCurrentDirectory can get complicated. If you look at the declare, you'll see it takes two parameters, nBufferLength and lpBuffer (a long and a string) and returns a long value.

```
Dim rVal As Long ' the return value
Dim sDir As String ' the current directory

sDir = Space$(255)
rVal = GetCurrentDirectory(255, sDir)
sDir = Left$(sDir, rVal)
```

In the code sample above, two variables are declared: rVal and sDir

rVal will hold the return value of the function (in this case it will be the length, in characters of the current directory)

sDir is initialized to 255 space characters in order to give the API function room to work with. sDir, when the function is finished, will hold the current directory. (And that's the tricky part, we are just passing in a large, empty string for the function to fill in.)

After the function completes, we trim off the excess right-hand spaces.

Conclusion:

These two examples barely scratch the surface of what the API does. The best reference I have seen for API functions is:

The Visual Basic Programmer's Guide to the Win32 API by Daniel Appleman

You have to know what the API can do before you can decide what you want to do with the API. Read as much as you can find out about these routines. Browse through the API viewer.

Remember that its not required you use these functions, Visual Basic has very highly optimized internal functions and methods which in many cases work better (i.e. faster) than their API equivalents.
- Robert Houghtaling

3. What is a callback?

The callback attribute declares a static callback function that exists on the client side of the distributed application. Callback functions provide a way for the server to execute code on the client.

The callback function is useful when the server must obtain information from the client. If server applications were supported on Windows 3.x, the server could make a call to a remote procedure on the Windows 3.x server to obtain the needed information. The callback function accomplishes the same purpose and lets the server query the client for information in the context of the original call.

Callbacks are special cases of remote calls that execute as part of a single thread. A callback is issued in the context of a remote call. Any remote procedure defined as part of the same interface as the static callback function can call the callback function.

Only the connection-oriented and local protocol sequences support the callback attribute. If an RPC interface uses a connectionless (datagram) protocol sequence, calls to procedures with the callback attribute will fail.

Handles cannot be used as parameters in callback functions. Because callbacks always execute in the context of a call, the binding handle used by the client to make the call to the server is also used as the binding handle from the server to the client.

Callbacks can nest to any depth.
- Guy Thompson