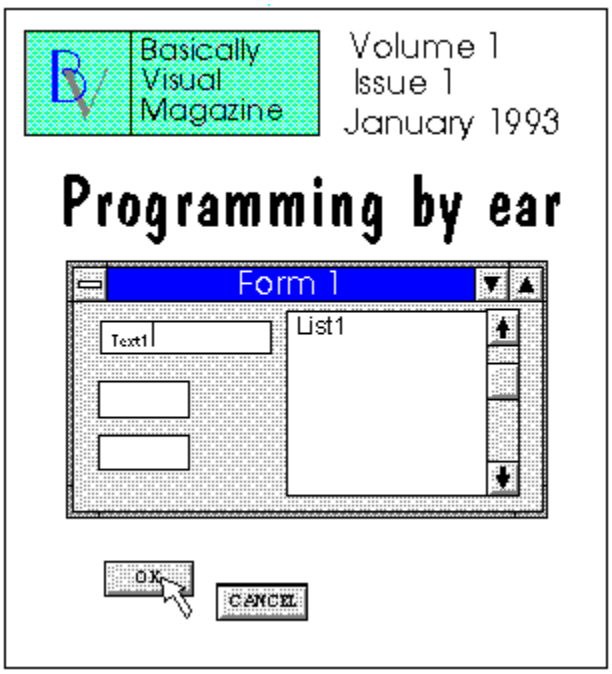



## This month:

---

**About the Authors**

---

**Tips, Tricks and Gotcha's**

---

**Next Month** in Basically Visual Magazine

---

**END SUB**

OPEN FILE

*by G.D. Walters*

Welcome to the premier issue of Basically Visual Magazine.

There was so much that I wanted to put into this issue, but so little time. Between my regular "8 to 5" job (that's what they tell me it's supposed to be anyway), the BBS, the magazine and trying to be a full-time father & husband, there's little time to do anything else. Please forgive me for allowing this issue to be so slim.

What can you expect to find from Basically Visual Magazine? Let me start answering that question by taking you back a couple of months. I originally envisioned a magazine that was based on Microsoft's Visual Basic programming language. As I tried to pull all the parts together, I realized that simply having a magazine that only covered one language, left a bunch of programmer's out in the cold. I also realized that the wave of the future will be the VISUAL programming platform. So, I decided that it would be better if I opened my horizons a bit and covered as much as I could. That's when I added support for Borland's Object Vision. Then I added Microsoft Access. So now, the magazine will be covering all of these languages and platforms in future articles.

With that said, here is the "grand scheme" for the future of the magazine, at least as far as my crystal ball will allow me to see. The magazine will be covering:

Microsoft Visual Basic for Windows

Microsoft Visual Basic for DOS

Borland Object Vision

Microsoft Access

Right now, I have two authors setup to write monthly columns, one for Visual Basic for DOS-Beginner Level and one for Visual Basic for Windows-Advanced level. As soon as I can get other authors, I want to have monthly columns for all four languages in both the beginner and advanced levels. As more languages start supporting the visual programming platform, we will setup authors for those, covering both levels. We will also bring other articles, tips, hints, bug reports, etc. With any luck, in a couple of months the size of the magazine will more than double.

So sit back and hold on. We are looking for some GREAT things happening in 1993.

# Visual Basic for Windows 2.0

By G.D. Walters

*SOFTWARE REVIEW*

*BV Rating:* 

In May 1991, Microsoft released Visual Basic for Windows 1.0. It was quickly taken up by both beginner and advanced level programmers as one of the fastest way to produce quality, bullet-proof Windows applications as well as prototypes. Training time was very fast also. For experienced programmers became proficient in Visual Basic in 2-3 weeks. There were, however, some major limitations in version 1.0. Even with the limitations, Visual Basic was embraced by corporations and software authors alike, and the number of applications and add-on tools flourished.

November 1992 came and Microsoft released version 2.0, a major upgrade. It is offered in two forms, the Standard edition and the Professional edition. So many things have been added and changed, that it's hard to know just what to talk about first. Rather than just stating a bunch of facts to start with, we'll jump right in and talk about the changes to the Visual Basic programming environment.

The first thing you will notice about 2.0, is that a new toolbar has been added under the menu. Figure 1 shows this toolbar.

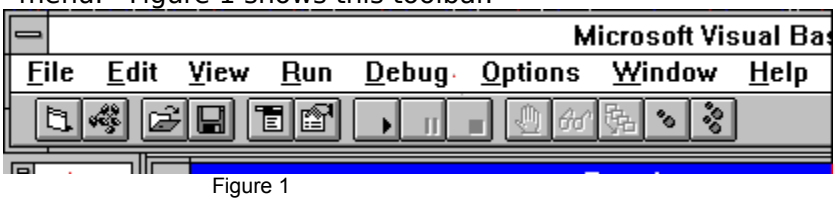



Figure 1

The buttons on this toolbar allows new forms and modules to be added, application debugging and stepping, starting, pausing and ending of the application as well as access to the menu designer.

Another addition to the environment is the properties window. This window allows faster access to the information about the form, single control or multiple controls. Figure 2 shows the properties window.

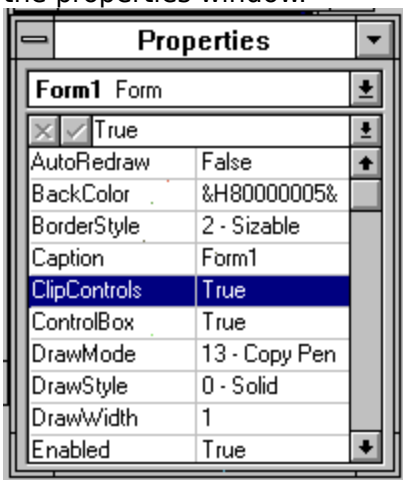



Figure 2

Did I say multiple controls? YES. Another enhancement of version 2.0 is the ability to select multiple controls by dragging a marquee box around the controls. When multiple controls are selected, the property window shows only the attributes that are common between all the controls.

There is also a new customizable AUTOLOAD.MAK file that contains any set of forms, controls or modules. This allows you to start VB with all your favorite tools available automatically.

There are many new goodies that make VB 2.0 a truly powerful application development language. One large complaint was that forms and *.MAK files were in a "tokenized" format. You couldn't export them to ASCII or read them directly. Version 2.0 keeps *.MAK files in standard ASCII format and Forms can be saved as ASCII files. Other improvements include Color-coded syntax in the code windows, enhanced debugging capabilities including breakpoints and a watch facility, a new Procedures dialog, smaller .EXE files and a MUCH improved application load and executing speed.

Stealing a bit from the press release kit that Microsoft provided, here is a quick breakdown of the differences between the Standard and Professional versions as well as new features for both.

| | **NEW FEATURES** | **FEATURES FROM 1.0 PRO** |
|---|---|---|
| VB 2.0 STANDARD | Speed and Capacity Improvements<br>Debugging<br>IDE improvements (property window)<br>Language Enhancements<br>MDI, Multiple Instances of Forms<br>Graphic Controls<br>Set-up Kit<br>ASCII Import/Export | Grid Control<br>OLE Client Control |
| VB 2.0 PROFESSIONAL | 2 MAPI Controls<br>Language-based ODBC Data Access<br>Comm Control<br>2 New Pen Controls<br>Masked Edit Control<br>SQL Server Driver for ODBC<br>Online Visual Design Guide<br>Enables third party CASE and SLM tools | Pen Controls<br>Multimedia Control<br>PicClip Control<br>Spin Button Control<br>3-D Buttons Control<br>Graph Control<br>Gauge Control<br>Animated Button Control<br>Key Status Control<br>Enhanced Common Dialog Control<br>Windows Help Compiler (Win 3.1)<br>Windows Set-up Kit<br>Online API Reference (Win 3.1)<br>PSS Knowledge Base<br>Control Development Kit |

The bottom line is, this re-work of Visual Basic is a winner. If you are wondering which version, professional or standard, you should buy, my suggestion is to get the professional version. You will see that it is more expensive, but the benefits far out weigh the cost.

The Programmer's Pit BBS

## The Programmer's Pit BBS System

Where can you ALWAYS find the latest issue of ***Basically Visual Magazine***? Where can you find ALL the back issues of ***Basically Visual Magazine***?

The Programmer's Pit BBS - Home of ***Basically Visual Magazine***, that's where.

**==========**

**(717) 845-2725 (717) 845-2725 (717) 845-2725 (717) 845-2725**

**==========**

We have 700+ MEG ONLINE and over 2 GIG off-line of applications, program source, games and information files. We are a member of FIDO and SPEAKEASY NETWORKS. We carry program source for:

**Visual Basic**
**Pascal**
**C / C++**
**Modula-2**
**Assembly**
**AND MORE!**

23 hours a day! 7 days a week!

If you have FREQ abilities, the magic filenames are:

| | |
|---|---|
| **FILES** | ALL ON-LINE FILES LIST |
| **NEWFILES** | ON-LINE FILES LESS THAN 15 DAYS OLD |
| **MAGAZINE** | THE LATEST ISSUE OF BASICALLY VISUAL MAGAZINE |
| **CD-ROM** | FILE LISTS OF CD-ROMS |

**ALL ON-LINE FILES CAN BE FREQ'ed.**
SEND FILE REQUESTS FOR OFF-LINE FILES.
***FIDONET*** ADDRESS - 1:270/612
***SPEAKEASY*** ADDRESS - 18:18/10

SUBSCRIPTION INFORMATION

## Subscription Information

***Basically Visual Magazine*** is published monthly and distributed via BBS Systems. ***Basically Visual Magazine*** is a shareware concept magazine. If you find the magazine to be useful, the price is $20.00 US. Funds per year.

To subscribe, please send check or money order to:

**BASICALLY VISUAL MAGAZINE**

**PO. BOX 1214**

**YORK, PA USA 17405-1214**

# VB/DOS KickStart

***by Anthony V. Seo***

Visual Basic for DOS - BEGINNER LEVEL

## The Beginning

The intent of this column is to help new users of Visual Basic for DOS (hereafter referred to as VB/DOS) get familiar with the IN's and out's of the package. If you are a long time Basic programmer, just making the move to the forms driven environment of VB/DOS or even a novice, starting out for the first time, this column hopefully will be of some interest to you. I am going to make the assumption that the readers at least have some familiarity with some of the Basic keywords.

Now I am sure that some of the old VB hacks out there will have 30 or 60 opinions as the rights and wrongs of what I am writing here, but I am trying to go through things that I feel should be learned at the beginning level. If you do have comments or suggestions, feel free to send them to me, care of this publication.

About myself, I have over 14 years experience in the personal computer marketplace. I first started with Basic on the TRS-80 Model I, and have worked with many different flavors and varieties over the years. In my current position, I run a systems consulting company, where we do custom application work, and also a software publishing company. All of my work is in the area of business applications and the tone of this column will be more towards that area, rather than games or graphics.

To get started, how do forms driven applications differ from traditional Basic programming? In the traditional Basic environment, each line of code is executed sequentially.

```
10 FOR X = 1 TO 100
20 Y = X * X
30 PRINT X, Y
40 NEXT X
```

When the program is run, the computer starts with line 10 and loops through the instructions in lines 20 and 30, then increments the counter X in line 40. Once X is greater than 100, then the program moves to next line. Since this is blank the program stops.

In the forms driven environment, when the program starts execution, nothing happens unless there is an Event to cause it to happen. This can be very confusing to someone (like myself), who is used to thinking in a standard sequential way of doing things.

When I write an application, once I design the actual data structures to be used, (and I will talk in detail about that in a later column), I layout the data entry flow and what validation is needed for each item. For instance, if it is a number, should it be between a range of numbers, or is zero or a negative number allowed? For example let us say that I would like the user to input a number to indicate a month. The code might look like this:

```
 1100 LOCATE 10, 10 : 'Position the cursor at a specific screen location
1110 INPUT X% : 'Tell Basic to accept keyboard input of a number only
1120 IF X% < 1 OR X% > 12 THEN GOTO 1100 : 'Test for a value between 1 to 12
```

Once I had a value between 1 and 12 the program would move onto the next line of code. Which is fine, except, suppose I wanted to allow the user to go back to a starting point, such as a menu without doing any data entry, or lets say once the data had been entered, I wanted to give the user a way of editing the data before saving it. This is where things get complicated. I could change line 1120 to say that if the value is outside the range of 1 to 12, that the program should go to line 1000, which might be my menu. That

would give the user away out, but every time they made a mistake, right back to menu to start over again. I could make a certain value as an "exit" number, such as 999 and test for it only. Or I could use a string variable and test for a certain key, example:

```
1100 LOCATE 10, 10 : 'Position the cursor at a specific screen location
1110 INPUT X$ : 'Tell Basic to accept keyboard input
1115 IF X$ = CHR$(27) THEN GOTO 1000: 'Test for the ESC key to allow an exit
1117 X% = VAL(X$): 'Convert the string into a number
1120 IF X% < 1 OR X% > 12 THEN GOTO 1100 : 'Test for a value between 1 to 12
```

Now this allows us to test for the Esc key, and then check for a valid number, but when you have 10 or 20 or these in a row, it still gets to be a pain to handle editing, once the data has been entered. This is where the forms driven environment makes application development a lot easier, plus it allows for more consistency between applications.

To continue with this example, let's design a form. Fire up VB/DOS then at the file menu select F for new Form. Call the form DATA1. You are then switched into the Form designer portion of VB/DOS. On the left side of your screen is a list called Tools, on the bottom is the Color Palate, and to the left and occupying most of the screen is a gray box titled DATA1. This is your new form. The tools to the right are your CONTROLS, and these are what we are going to use to build our screen with. (NOTE: I am going to use the default control name structure here to avoid confusion.)

Using your mouse, goto the Tools list and select the one marked Command Btn. A Command Button is using to active a set of procedures that you define. I use them in place of a menu. A box should appear on your form that says Command1. You can move that box anywhere on the form using the mouse so for now put it at the bottom. On the menu area above the form there is a line that says Property [ some text] and a down arrow. When you click on the down arrow a list drops down. Move through the list until you see the one marked Caption, then select it. You will then notice that the word Command1 appears in the brackets to the right. Click there with your mouse and you can edit the name. For this example change it to Add. The name in the box on the form should also change to Add. If you want to experiment, you can change the size of the box, and also it's color. Just make sure that the box is selected before going to the Properties or Color Palette. The term here to remember is that when an object is selected it now has FOCUS. FOCUS is a very important word in the VB environment. Once you are done with the Command1 button, go to the Tools menu and select Command Btn again. This time it will show up with the caption of Command2. Change the caption on this button to Exit.

Now again using your mouse goto to the Tools list and select the tool called Text Box. A textbox is where you do most of you data entry work. This time a box will appear with the work Text1 in it. Again using the Properties menu, find the line that says Text, but this time just delete the word Text1. SHORTCUT #1: When you select the text Property, hitting your Backspace key will erase the contents of the line. The same works on Caption lines as well.

Now press your F12 (Function 12 key). You should get prompted with a message to "Exit to the Programming environment". Hit Yes, you will get prompted with the message "Project or source files have changed. Save them now ?". Press Yes. If the file already exists, you will also get asked if it is okay to overwrite it, again yes. You will then see a screen called Event Procedures. The first column is a list of files, in this case it should only be DATA1.FRM, the second column is Objects and should only contain the lines Form (Data1), Command1, Command2, and Text1. The third column is called Events, and this list changes with the type of object that is selected. First use your mouse to highlight the Command1 object, then click twice on the Click event. You are then at a screen that is titled DATA1.FRM:Command1_Click. The top line of code should say Sub Command1_Click (). Right below that line type the following; Text1.SETFOCUS. The screen should show as follows:

```
Sub Command1_Click
    Text1.SETFOCUS
End SUB
```

What this does is to tell the program that when you click on the Add box, that the FOCUS should be moved to the Text1 box. Then press F12 again and do the same thing for Command2, except the code is as follows:

```
Sub Command2_Click
    End
End SUB
```

This gives us away to stop the program, without having to fight with the CTRL-BREAK key.

Then press F12 again, highlight the Text1 Property and select the Event labeled LostFocus. Your screen will then show a subroutine tiled Text1_LostFocus (). This is where we want to do our validity checking. Now anything that gets typed into a Text Box is text, so to do numeric checking, we must convert to a numeric value. We are also going to use a neat little function of VB/DOS called MsgBox to tell us if our data is wrong, so under the Sub Text1_LostFocus () type in the following:

```
SUB Text1_LostFocus ()
IF Text1.Text = "" THEN
    oops% = MSGBOX("Month not entered", 5, "Warning")
        SELECT CASE oops%
        CASE 4
            Text1.SETFOCUS
        CASE 2
            Command1.SETFOCUS
        END SELECT
END IF
If VAL(Text1.Text) < 1 Or Val(Text1.Text) > 12 Then
    oops% = MsgBox("Invalid Month entered", 5, "Warning")
        SELECT CASE oops%
        CASE 4
            Text1.Text= ""
        Text1.SETFOCUS
        CASE 2
            Command1.SETFOCUS
        END SELECT
END IF
Command2.SETFOCUS
END SUB
```

When everything is done, always save your work before running the program. Once in a while you hit some undocumented feature, that can cause you to loose lots of work if the machine hangs. Then press the F5 (Function 5) key to start the program. If you get any errors, double check your typing.

Now what is going on here is that when you press the Add button (Command1) the cursor jumps up to the Text1 box. You can then type in a number or hit enter. Then use the TAB key or the mouse to jump (at this point) to the Exit Button. If you have entered a

number between 1 and 12, then nothing should happen except that the Exit button now has the FOCUS. If you just tabbed without entering anything then the "Month Not Entered" message will appear. Depending on which button you select (Retry or Cancel), you will be sent back to Add or to the Text1 box. If you entered a number that was greater than 12 or less than 1 the same box would appear, except that the message would be "Invalid Month Entered", with the same options, except that if you hit Retry, the first value you entered will be cleared. In a later column I will spend some time on the MSGBOX function.

You can experiment with different tests for numeric or text strings. Text1.Text is a string value, and all of the standard Basic string functions, such as LEFT$, RIGHT$, MID$, as well as the QuickBasic UCASE$, LCASE$, LTRIM$ and RTRIM$ functions. Next column we will work with some of the other Objects available and Controls that are available. Till the next time.

# Visual Basic Depot

***by Mark Wisecarver***

Visual Basic for Windows Programming - ADVANCED

### 1993: Everything's new

Welcome to an all new era in Windows programming. Hi gang, my name is Mark Wisecarver and I am going to be bringing you this column each month tackling the world of GUI programming with Visual Basic.

I am a senior programmer for a major corporation and currently develop some where in the neighborhood of 10-30 VB applications each week.

If VB programming for MS Windows is your forte, then this column will benefit you each month. The intention will be to try and keep you above everything else interested and informed.

So what's new with VB? As all of you by now know version 2.0 has proved to be everything Microsoft had promised it would be. There isn't a faster way to create a Windows application than Visual Basic and now with v2.0 you can do the job with a lot more flexibility. I will be telling you about many of the nice little additions and tricks over the next few issues of this Magazine and many of them are nothing short of impressive. With this release the public is once again given the choice of buying the Standard Edition or the slightly more expensive, (But well worth the money), Professional Edition.

Each month I will answer at least one readers question and this months first question comes from "AK" of Dallas Texas.

AK: Mark, how do I force Windows to display a hidden form?

MW: AK, The Windows API library has a number of functions that will force Windows to display a hidden window. This project as an example demonstrates how a previously hidden window can be displayed under program control.

```
' Declare these constants and Windows API declarations.
Global Const SW_SHOWNA = 8
Declare Sub BringWindowToTop Lib "User" (ByVal hWnd As Integer)
Declare Sub ShowWindow Lib "User" (ByVal hWnd As Integer, ByVal
nCmdShow As Integer)
' Put the following code in the Form_Load event subroutine.
Sub Form_Load ()
' Make Label1 fill entire form.
 Label1.Top = 0
 Label1.Left = 0
 Label1.Width = ScaleWidth
 Label1.Height = ScaleHeight
' Position form in Lower right corner of screen.
 Form1.Top = Screen.Height - Form1.Height
 Form1.Left = Screen.Widht - Form1.Width
' Set up crosshair mouse.
```

```
 Form1.MousePointer = 2
End Sub
' Put this code in the Timer1_Timer event subroutine. (Interval 1000)
Sub Timer1_Timer ()
 ShowWindow Form1.hWnd, SW_SHOWNA
 WindowState = 0
 Label1.Caption = Time$
' Just for the heck of it put this code in the Label1_Click event.
Sub Label1_Click ()
  End
End Sub
```

***Comments***

The ShowWindow API call displays the specified window but doesn't make it active. To activate the window, simply click on the form's caption bar.

The BringWindowToTop API call both displays the window and activates it in one step. The declaration for this function is included in the Global code.

You may want to set Form1's BorderStyle to 0 (None), so that is will take up as little space on the screen as possible.

The form's MousePointer is set to a crosshair as a further indication of when the form is active.

R.D. of Ida Mi: Is there an easy way to find the difference between two times?

M.W.: There sure is. First let's create a new form with one Command button, 2 Timers, and two Labels. You can add more to spruce it up as you like but that's what you'll need for these examples.

---

For the first example let's make a simple 12 hour format clock.

```
Sub Timer1_Timer ()
    Label1.Caption = Format$(Now, "hh:mm:ss: am/pm")
End Sub
```

The first example was very easy wasn't it? Make sure that you set the Interval for Timer1 to 1000. (1000 = 1 second) You can make the Interval anything you want but for this example 1000 is best.

Now we are going to use the Command button to give the user a chance to find the difference between two user input times.

```
Sub Command1_Click ()
    Msg$ = "Please Enter the first of two times that you "
    Msg$ = Msg$ + "would like to find the hours, minutes, "
    Msg$ = Msg$ + "and seconds that Lie between."
    Title$ = "First Time"
     EnteredTime$ = InputBox$(Msg$, Title$, Time$)
    BeginTime = TimeValue(EnteredTime$)
    BH = Hour(BeginTime)
    BM = Minute(BeginTime)
     BS = Second(BeginTime)
```

```
    Msg$ = "Please Enter the second of the two times that you "
    Msg$ = Msg$ + "would like to find the hours, minutes, "
    Msg$ = Msg$ + "amd seconds that Lie between."
    Title$ = "Second Time"
    EnteredTime$ = InputBox$(Msg$, Title$, Time$)
    EndTime = TimeValue(EnteredTime$)
    EH = Hour(EndTime)
    EM = Minute(EndTime)
    ES = Second(EndTime)
     Difference = TimeSerial(Abs(BH - EH), Abs(BM - EM), Abs(BS - ES))
    MsgBox "The difference between these times is " + Format$(Difference, "hh:mm:ss")
End Sub
```

In that example we are giving the user the time difference between two times. By default we are allowing the InputBox to use the current system time unless the user inputs anything different.

Now for my last example I will show you how to design a clock that counts backwards. Let's say for this example that you would like to show the hours, minutes, and seconds that lie between the current time and a coded time. For this example we are going to use 12:00 Midnight.

***NOTE: Remember to set Timer2's Interval to 1000***

```
Sub Timer2_Timer ()
    EnteredTime$ = Format$(Now, "hh:mm:ss am/pm")
    BeginTime = TimeValue(EnteredTime$)
    BH = Hour(BeginTime)
    BM = Minute(BeginTime)
    BS = Second(BeginTime)
    EnteredTime$ = "11:59:59 pm"
    EndTime = TimeValue(EnteredTime$)
    EH = Hour(EndTime)
    EM = Minute(EndTime)
    ES = Second(EndTime)
    Difference = TimeSerial(Abs(BH - EH), Abs(BM - EM), Abs(BS - ES))
    Label2.Caption = Format$(Difference, "hh:mm:ss")
End Sub
```

I have given you a few examples here of how to display time in different formats and there remain many more ways to use Time$ and Date$ formats in Visual Basic. If you have any more questions we encourage you to write in with them along with your comments.

Until next time I wish all of you Happy programming and look forward to the next issue.

Mark Wisecarver

Send your questions and comments to this magazine care of VB Depot.

## About the Authors

### *G.D. Walters*

Greg Walters has been working with computers and programming since 1972. He has worked as a freelance programmer and a MIS Director. He has written articles for Modules & Definitions Magazine and has worked as a freelance technical editor for Osborne/McGraw-Hill on a number of books. He is currently a programmer for St. Onge Company in York, PA. He is also the editor of Basically Visual Magazine and runs The Programmer's Pit BBS.

### *Mark Wisecarver*

Mark is a senior programmer for a major corporation and lives in Flat Rock, Michigan . He has been a programmer since 1979. He has background in Basic, GWBasic , Quick Basic, Microsoft C, Assembly Language, JCL, Turbo Pascal and Visual Basic programming. He runs an online service free to all specializing in Visual Basic code.

### *Anthony V. Seo*

Tony has been working in the personal computer area for 14+ years. He has a background in Basic and Xenix/Unix programming. Tony runs Marketing & Automation Resource Company, a systems consulting company and software publishing company in Camp Hill, PA. He also runs the M.A.R.C. Information System BBS System.

## TRADEMARKS

## Next Month in Basically Visual Magazine

Borland's Object Vision PRO - REVIEW
Microsoft Visual Basic for DOS - REVIEW
More from Mark & Tony & Greg
Diskette Labeling Program in VB-Win
---------- AND MORE !! ----------

# END SUB()

*Closing Thoughts from Greg*

Well, we've come to the end of the first issue. As I said before, it's rather short. In fact, it contains only about a third of what I ***REALLY*** wanted to put in. Next month will be about the same size, since I only have 15 days to pull it all together.

A couple of people have asked me about the future of the magazine. That depends on you. If you read **and** subscribe, I can afford to do more with it. I've also had a number of people comment that they don't really think they could write an article, that they don't know enough. Well here is your chance. If you find a neat trick or a bit of code that you have written, send it in. You only have to write enough to document the file so that others can understand it. We'll do the rest.

We have set up a message echo to support the magazine. Here is a list of people and their network addresses to contact if you would like to leave a comment, send code or articles...

| BOARD | ADDRESS | SYSOP | PHONE NUMBER |
|---|---|---|---|
| **The Programmer's Pit BBS** | FIDONET 1:270/612 | Greg Walters | (717) 845-2725 |
| | SpeakEasy 18:18/20 | | |
| **M.A.R.C. Information Systems** | FIDONET 1:270/711 | Anthony Seo | (717) 763-4419 |
| **Light House** | FIDONET 1:2380/410 | Mark Wisecarver | (313) 379-3945 |
| **Zack's Shack** | FIDONET 1:367/641 | Zack Jones | (210) 653-2115 |
| **EMC/80** | FIDONET 1:100/555 | Jim Harre | (314) 843-0001 |
| **Part II** | SpeakEasy 18:18/1 | Elwood Henny | (717) 864-7721 |
| **The Computer Den** | SpeakEasy 18:18/10 | Mark Humbert | (717) 252-0553 |

Of course, you can find the magazine on these points also. As more points join the echo, we'll let you know.

For now, have a GREAT 1993. We'll see you in February.

Greg

# M.A.R.C. Information Systems

The ***M.A.R.C. Information System*** is run as a service to the clients and guests of Marketing & Automation Resource Company of Camp Hill, PA.

We are a systems consulting and integration company, specializing in MS/DOS and Xenix/Unix accounting systems. We offer custom software, system support, training, and a complete line of hardware including many specialty items. This is also the home of Master\sofT Software Publishing, specializing in engineering and scientific software for the PC.

The BBS hours are from 6:00PM to 8:00 AM Monday - Friday and 6:00 PM Friday and 8:00 AM Monday. The BBS is up during the daytime hours as well, but this is not on a fixed schedule. We are Fido Node 1:270/711. We carry the following FidoNet Conferences:

| | | |
|---|---|---|
| ***BUSINESS*** | ***CONSULTING*** | ***LAW*** |
| ***HOME_OFFICE*** | ***STOCK_MARKET*** | ***QUICK_BASIC*** |
| ***UNIX*** | ***VISUAL_BASIC*** | ***PSD_BASIC*** |
| | ***PC_CONSULTING*** | |

Our voice phone number is 1-717-763-5772.

Thank you for calling.

Anthony Seo

# LightHouse File System

**The Light House BBS**
**(313)379-3945** USR HST 14.4
WILDCAT! v3.55M Reg# 92-2226M
**(313)379-3995** v.32 v.42
FIDONET NETmail address 1:2380/410
CompuServe: 72400,505

* **Visual Basic** programmers support BBS (File and Message areas)
* **PKWARE** Distribution License (Upgrades on-line within 24 hours)
* Over **15,000** files on-line
* No charges, or donations for access or membership

Flat rock, Michigan US. of America

# *BBS Watch*

This area is to let you know what is coming out on the bulletin boards and where you can get it.

As new code and software is released using Visual Basic, Object Vision or Access, we will include it here.

This month we'll let you know about VBFE. This is the Visual Basic File Echo on FIDONET. New Visual Basic source code files are sent out on the file echo automatically to it's distribution points. The system operators of you local FIDONET board can get these files by polling one of the distribution points. As more and more boards poll, VBFE can get on the "backbone" and more people will have easy access to the network. So if your local FIDO board doesn't carry VBFE, let them know you want it.

Here are the current distribution points for VBFE...

| | | |
|---|---|---|
| EMC/80 | (314) 843-0001 | 1:100/555 |
| LIGHTHOUSE | (313) 379-3945 | 1:2380/410 |
| PROGRAMMER'S PIT | (717) 845-2725 | 1:270/612 |
| WRITE BYE NIGHT | MAIL ONLY | 1:142/928 |

I should take a moment to talk about the FIDONET Visual Basic Echo. If you want to find out ANYTHING about Visual Basic, talk to the folks up there. They are very helpful. Special thanks to Margaret Romao for her work as echo moderator and allowing me to beg for authors and intrest. Visual Basic echo is available on the FIDONET Message Echo Backbone.

More next Month!

# Beginning Basics

Visual Basic for Windows - Beginner's Level

*By Greg Walters*

Programming by Ear

Since this is our first time together, I really don't know just how much a beginner you are. So, I'm going to make a few assumptions...

1) You have Windows 3.1.

2) You have Visual Basic for Windows.

3) That you know how to use Windows.

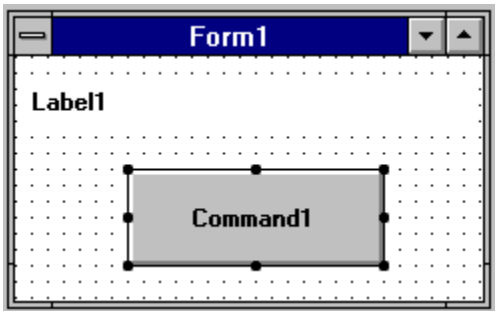
4) That you have a TINY bit of knowledge about Basic programming.

If I'm wrong, please let me know. This column is to help you, not confuse you.

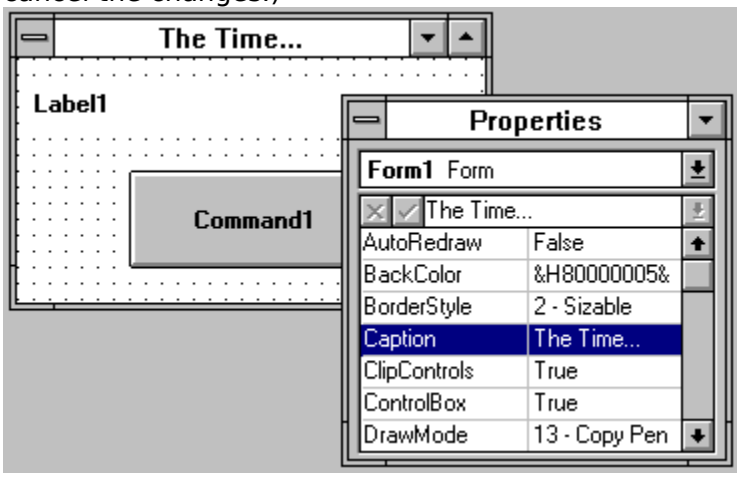
I'm going to go kind of fast this time, so hold on.

Our first program will be a VERY simple clock. There will be a box with a button that says "The Current Time". When the user clicks on it, the time appears on the screen.

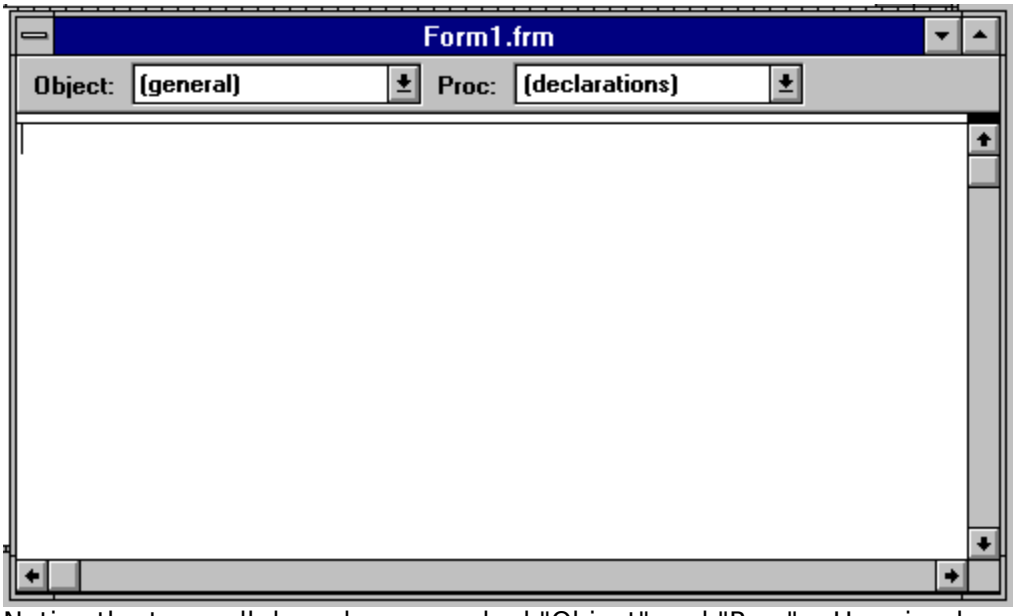
First thing to do, is select the label box tool and place it on the form. A quick way to do this is to double click on the label box button. This puts a default sized and placed label box. (This trick works on any of the tools.) Move the label box to the top of the form. Now double click on the command button. Resize the form until it looks something like this...




Now we want to change the looks of our program a little bit. Click on the form itself (not one of the objects in the form). This will call up the properties window. Make sure it says "FORM1 Form" in the top box. Now go to the bottom box and find the "Caption" attribute. Change this to say "The Time..." by clicking on "Caption" and going to the text box above it. Once you have changed it, click on the check mark to save your changes. (The "X" is to cancel the changes.)



Now, using the same method, change the label caption to nothing (Erase the text in the box), set the label border style to "Fixed Single", and change the caption of the command button to "The Current Time". Open the code window by clicking on the code button in the project window.
Here is what the code box looks like...




Notice the two pull down boxes marked "Object" and "Proc". Here is where we assign our code routines.

If we were going to write a subroutine called "time1", then we would type

SUB Time1()

and press the <Enter> key. This moves our subroutine declaration to a new editing window and places the subroutine into the "Proc" list pull down. Visual Basic also adds the ending to our subroutine for us.

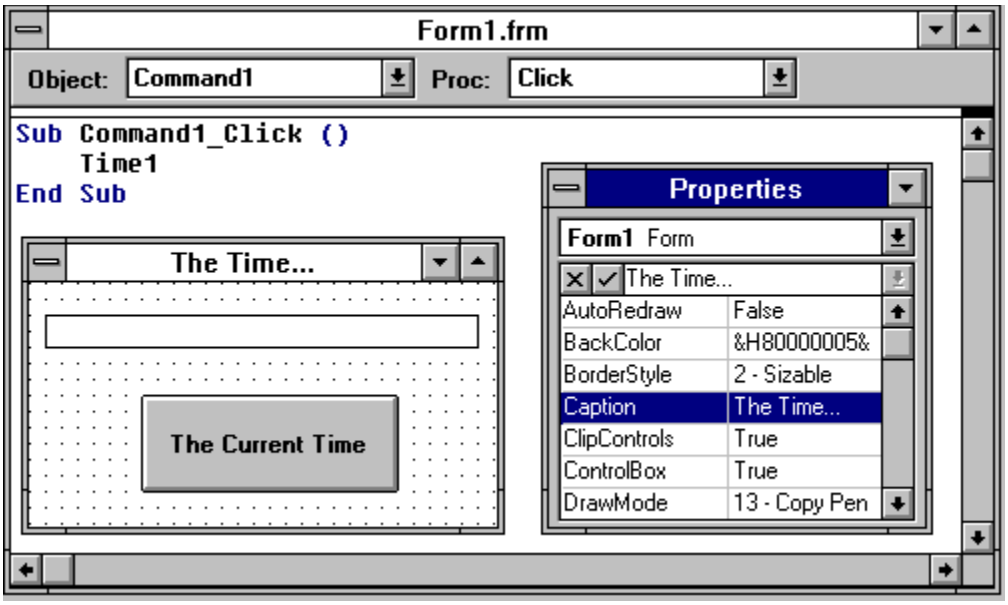
```
Sub Time1 ()

End Sub
```

Now all we have to do is fill in our code between the two lines.

```
    Label1.Caption = "The current time is "+ time$
```

Our subroutine Time1() has to be called, or it never gets used. We will use the Command1 button that we placed on the form earlier to call it. I've placed the form and property window on top of the code window so you can see all three at once.




We want Time1() to be called when ever the user clicks the mouse on the command button. To do this, we insert the call to the subroutine in the Command1_Click() method. The call is simply...

Time1

(We don't need the '()' since we don't have anything to send in. We'll cover that next month.)

That's it. Now when the program is run, it waits for the user to click on the command button. Once this happens, Windows is notified that a mouse click occurred in the "The Time..." window, and it occurred on our command button. Now the button gets the message and calls our "Time1" subroutine, which in turns sends the text into the label box. It's a lot harder to explain what is happening, than to actually make it happen.

Well, that's it for this month. Remember, the easiest way to learn is to play...So PLAY!

Greg

Time$ is a function in Visual Basic that returns the current system time as a string.


This is the label button.


This is the command button.

# Tips, Tricks and Gotcha's

PROGRAMMERS TIP (From Mark Wisecarver)

There is no setting for controlling the use of expanded memory. Remember that in Standard mode, applications must fend for themselves to access expanded memory. In Enhanced mode, Windows simulates expanded memory using extended memory. Windows doesn't do expanded memory any more.

---

Reprinted with permission from Visual Basic Knowledge Base

Permission granted by Microsoft Corp.

How to Close VB Combo Box with ENTER Key

Article Number: Q84474

If you open a combo box and then use the ARROW keys to scroll through it, pressing the ENTER key will not close the combo box like a mouse click will. This is normal behavior. The following example demonstrates how to make a combo box close when the ENTER key is pressed.

More Information:

The following program makes use of the Windows API SendMessage function to send the combo box the

message to close. This is done only after the ENTER key is detected in the KeyPress event for the combo box.

Two Windows API Declare statements must be added to your application. These can be added to the general Declarations section of the form containing the combo box.

Steps to Reproduce Behavior

1. Run Visual Basic, or from the File menu, choose New Project (ALT, F, N) if Visual Basic is already running. Form1 is created by default.

2. Add the following two declarations to the general Declarations:

```
    Declare Function SendMessage% Lib "user" (ByVal hWnd%, ByVal wMsg%, ByVal wParam
%,
        ByVal lParam&)
    Declare Function GetFocus Lib "user" () As Integer
```

(Note that the first Declare must be on just one line, not split across two lines as it is

here.)

3. Place a combo box on Form1.

4. Under the KeyPress event for the combo box, place the following code:

```
If KeyAscii = 13 Then
    Const WM_USER = &h400
    Const   CB_SHOWDROPDOWN = WM_USER + 15

    Combo1.SetFocus
    BoxwHND% = GetFocus()
    r& = SendMessage(BoxwHND%, CB_SHOWDROPDOWN, 0, 0)
    KeyAscii = 0
End If
```

5. Place a command button on Form1.

6. In the Click event for Command1, place the following code:

```
' This will add some data to the combo box.
for i =1 to 10
   Combo1.AddItem i
Next i
```

7. Press F5 to run the application.

8. Choose the Command1 button to fill the combo box.

9. Open the combo box with the mouse, and scroll down with the ARROW keys.

   Pressing the ENTER key will close the Combo Box.