Microsoft Visual Basic for Windows: Bugs, Fixes, & Updates

Prepared 04/20/95

Microsoft Visual Basic for Windows: Bugs, Fixes, & Updates

Unfixed Bugs

BUG: TABs Paste Incorrectly as | to VB.EXE's Immediate Window
Article ID: Q73700

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

TAB characters may be (from the Windows Clipboard) incorrectly PASTEd into the Immediate Window in Visual Basic. In Visual Basic, version 1.0, TAB characters may be incorrectly PASTEd in as pipe [|] symbols. In Visual Basic, version 2.0, TAB characters may be incorrectly PASTEd in as '\177', which looks like a small black box.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic programming system versions 1.0 and 2.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. From Windows, version 3.0, run NOTEPAD.EXE and enter the following text:

   ```
   a <TAB> a
   ```

   This will be displayed in the following format:

   ```
   a     a
   ```

2. Select the text with the mouse and choose Copy from Notepad's Edit menu to copy the text to the Windows Clipboard.

3. Start Visual Basic and press F5 to run the blank program (or from the Run menu, choose Start).

4. Break the program by pressing CTRL+BREAK, then click the mouse on the Immediate Window.

5. Press SHIFT+INSERT to enter the selected text into the Immediate Window. Observe that the Immediate Window incorrectly displays the following text:

   ```
   a|a
   ```

```
   instead of displaying the following:

   a     a

   NOTE: A pipe symbol is displayed in version 1.0, however in version 2.0,
   '\177' is displayed, which looks like a small black box.

6. If you end the program (by choosing the End command from the Run
   menu), you will be able to successfully PASTE (SHIFT+INSERT) the
   correct text into any code window:

      a     a
```

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Scroll Box Flashing Not Updated If Bar Resized w/ Focus
Article ID: Q73839

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, versions 1.0, 2.0, and 3.0
- Microsoft Windows, versions 3.0 and 3.1

SYMPTOMS
========

There is a Microsoft Windows version 3.x (3.0 and 3.1) problem updating the flashing indicator for the scroll box on a vertical scroll bar. This Windows problem affects vertical scroll bars in Microsoft Visual Basic programming system for Windows. This article describes how to work around this problem.

WORKAROUND
==========

You can work around this problem by doing the following:

1. In step 2 of the More Information section, add additional code so that the Form_Click procedure appears as follows:

```
   Sub Form_Click ()
      Const True = -1, False = 0
      VScroll1.Height = VScroll1.Height * 2
      VScroll1.Enabled = False
      VScroll1.Enabled = True
   End Sub
```

2. Follow the directions for steps 3, 4, and 5 in the More Information section. You should notice that the problem no longer exists. The flashing has been updated correctly in the same position as the scroll box.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Windows versions 3.0 and 3.1. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic and place a vertical scroll bar on a form.

2. Place the following code in the Form_Click event procedure:

```
   Sub Form_Click ()
      VScroll1.Height = VScroll1.Height * 2
   End Sub
```

3. From the Run menu, choose Start, or press F5 to run the example.

4. Click and drag the flashing scroll box (on the scroll bar) to the middle (down from the top).

5. Click the form to execute the Form_Click procedure, which doubles the height of the scroll bar. Observe that the scroll box correctly moved to the middle of the longer scroll bar, but the flashing indicator failed to also move.

Additional reference words: buglist3.00 buglist3.10 1.00 2.00 3.00 3.10
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: [Character May Sort Incorrectly in List or Combo Box
Article ID: Q74132

```
---------------------------------------------------------------------
The information in this article applies to:

 - Microsoft Visual Basic programming system for Windows, version 1.0
 - Microsoft Windows, versions 3.0 and 3.1
---------------------------------------------------------------------
```

SYMPTOMS
========

An example below demonstrates a problem using the Sorted property to sort a string beginning with a bracket ([) in either a list box or combo box in Microsoft Visual Basic version 1.0 for Windows.

STATUS
======

This sorting problem is caused by Microsoft Windows versions 3.0 or 3.1, not by Visual Basic. Microsoft is researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. In the Visual Basic environment, choose New Project from the File menu.

2. Place two list boxes or two combo boxes on the form.

3. From the Properties Bar, set the Sorted property for either the two list boxes or two combo boxes to True.

   Note: Do not invoke List1.Sorted = -1 within the code of an event procedure because this causes the run-time error "'Sorted' property cannot be set at run time."

4. Now add some code to the Form_Click event procedure. Below are two separate examples of the code to add depending on if you are using list boxes or combo boxes:

```
   Sub Form_Click ()          Sub Form_Click ()
      List1.AddItem "["          Combo1.AddItem "["
      List1.AddItem "\"          Combo1.AddItem "\"
      List1.AddItem "a"          Combo1.AddItem "a"

      List2.AddItem "a"          Combo2.AddItem "a"
      List2.AddItem "\"          Combo2.AddItem "\"
      List2.AddItem "["          Combo2.AddItem "["
   End Sub                    End Sub
```

5. Run the code by pressing the F5 function key or choosing Start from
   the Run menu.

6. Click the form to see the sequence "a [\" in the first list box
   or combo box and to see the different sequence "[\ a" in the
   second list box or combo box.

This reveals an inconsistency with an internal Windows 3.0 sorting
routine. If you replace the character "[" with the character "b", the
two boxes correctly sort in the same order: "\ a b". The problem is
with sorting the "[" character.

Additional reference words: noupd buglist3.00 buglist3.10 1.00 3.00 3.10
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Can Click in Code Window Without Activating it in VB.EXE

Article ID: Q74194

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you have both a form and code window present at design time in Microsoft Visual Basic with the current focus on the form, clicking the upper or lower edge of the splitter bar in the code window fails to shift the focus to the code window. Clicking anywhere else in the code window correctly shifts the focus and activates the code window.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

To reproduce this problem in Visual Basic, choose the New Project option from Visual Basic's File menu. Double-click Form1 to open a code window, then click Form1 to return focus to the form. Now place the tip of the mouse pointer on the upper or lower edge of the code window's splitter bar such that the pointer remains an arrow, and is not an I-beam pointer or splitter pointer. Clicking now fails to shift the focus to the code window. You can click anywhere else in the code window and the code window will correctly become the active window.

Note that the "splitter bar" (the horizontal border just above the editing area and just above the vertical scroll bar) allows you to split the code window into two parts, which allows you to view two different sections of code at once.

Additional reference words: buglist1.00 buglist3.00 1.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Pressing ESC or CTRL+BREAK Makes Mouse Pointer Disappear
Article ID: Q74409

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

In Microsoft Visual Basic, the mouse pointer fails to be displayed if you press ESC or CTRL+BREAK within the Code window. (The mouse pointer correctly reappears if you move the mouse.)

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic with a New Project.

2. Double-click the form to bring up the Code window. Notice the I-beam mouse pointer within the Code window. If you place the mouse pointer outside of the Code window, you will see the I-beam change to an arrow.

3. Place the mouse pointer back in the Code window to display the I-beam pointer. Press either the ESC key or the key combination CTRL+BREAK. The I-beam mouse pointer will temporarily disappear.

4. The pointer will be redisplayed if you move the mouse.

Additional reference words: buglist1.00 buglist3.00 1.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtRun

BUG: No Beep When Click Form and the Menu Design Window Is Up

Article ID: Q74518

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The Menu Design window, used to create pull-down menus for Visual Basic forms, is a modal dialog box. Therefore, clicking any other Visual Basic window when the Menu Design window is visible should fail to transfer the focus and should generate a beep to notify you that you cannot act outside the dialog box. However, in Windows version 3.0, there is no beep when you click the Visual Basic form.

All other Visual Basic windows, such as the ToolBox, Color Palette, Project Window, and the main Visual Basic menu bar all respond with a beep when the Menu Design window is active -- as they should. In all cases, focus is maintained by the Menu Design window -- as it should be.

RESOLUTION
==========

This problem does not occur in Windows version 3.1.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article when using Windows version 3.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist1.00 buglist2.00 1.00 2.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Incorrectly Accessing System Menu of Hidden Form
Article ID: Q74564

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

It is possible under certain circumstances to incorrectly access the system menu of a hidden form in Visual Basic.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic, or choose New Project from the File menu.

2. Set the WindowState property of Form1 to 1 (minimized).

3. Enter the following line of code in the Form_Resize event
   procedure of Form1:

```
      If WindowState = 2 Then Hide    'WindowState 2 = maximized
```

4. From the Run menu, choose Start.

5. Click the Form1 icon to bring up the system menu for Form1.

6. From the the Form1 system menu, choose Maximize. Form1 will maximize
   and then hide.

7. Press ALT+SPACE to activate the Form1 system menu.

A system menu will appear in the upper left corner of the screen, even though Form1 is hidden.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbui kbprg kbbuglist
KBSubcategory: APrgWindow

BUG: Duplicate PostScript Font Names in VB Printer.Fonts List

Article ID: Q75092

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When a PostScript printer driver is active in Microsoft Windows version 3.0, the Fonts(index%) property of Visual Basic's Printer object may return one or more duplicate font names at run time. This will not occur in either Visual Basic version 1.0 or 2.0 if you are using Microsoft Windows version 3.1.

CAUSE

This problem is caused by Microsoft Windows version 3.0 itself, not by Microsoft Visual Basic.

STATUS

Microsoft has confirmed this to be a problem with Microsoft Windows version 3.0. The problem was corrected in Microsoft Windows version 3.1.

MORE INFORMATION

The following program displays the list of font names available for the PostScript printer currently selected in the Windows Control Panel:

```
   Sub Form_Click ()
      For J% = 0 to Printer.FontCount - 1
      Print Printer.Fonts(J%)
      Next J%
   End Sub
```

In some cases, when a PostScript printer is active in Windows, one or more fonts are listed twice.

Additional reference words: 1.00 2.00
KBCategory: kbprint kbprg kbcode kbbuglist
KBSubcategory: APrgPrint

BUG: ExtFloodFill Won't Fill Over QBColors If AutoRedraw=True
Article ID: Q75640

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you try to use the ExtFloodFill() API function in Windows version
3.0 or 3.1 along with the QBColor() function that is included in Visual
Basic, the first eight colors are displayed incorrectly on some
computers.

CAUSE
=====

With some computers, this problem causes the Fill Tool of the Iconworks
sample application provided with Microsoft Visual Basic to fail when
attempting to fill over QBColors (1-8).

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article with Microsoft Windows versions 3.0 and
3.1. We are researching this problem and will post new information here in
the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic and begin a new project.

2. Place a picture box on the Form. In the Properties bar for the picture
   box, set the AutoRedraw property to True and the FillStyle property to
   Solid.

3. Place the the following code in the General Declarations section of
   the code window for Form1, and enter the entire Declare statement on
   one, single line:

```
   DefInt A-Z
   Declare Function ExtFloodFill% Lib "GDI" (ByVal hdc, ByVal x, ByVal y,
                               ByVal crcolor as Long, ByVal wfilltype)
```

4. Place the following code in the Form_Click event procedure:

```
   Sub Form_Click ()
```

```
      Static I
      I= I + 1
      Picture1.BackColor = QBColor(I)
      x = ExtFloodFill(Picture1.hdc, 1, 1, Picture1.BackColor, 1)
      Print I;x
      Picture1.Refresh
   End Sub
```

5. Run the sample by pressing the F5 key. Notice that various colors are incorrectly displayed for QBColors 1-8 and that the return value from ExtFloodFill, held in x, is 0. QBColors 1-8 should be displaying black and the value for x should equal 1, not 0. QBColors 9-15 are correctly displayed.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbgraphic kbprg kbbuglist
KBSubcategory: APrgGrap

BUG: Duplicate Procedure Name Alters Original Capitalization

Article ID: Q76514

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you enter a Sub statement with a procedure name that duplicates an existing procedure name in spelling but not in capitalization, you will receive a "duplicate definition" error message, but the original procedure name will be changed to match the new capitalization. This also happens if you choose New Procedure from the Code menu and enter a duplicate name.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic versions 1.0, 2.0, and 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

If you choose the OK button after receiving the "duplicate definition" error message, the old name of the subroutine will be updated to show the changes in capitalization, even though there was a "duplicate definition" message.

Steps to Reproduce Problem

Method 1
----

1. Within a module, create a Sub procedure called "a".

2. From the Code menu, choose New Procedure. Name the procedure "A" and choose the OK button when the "duplicate definition" message is displayed.

Method 2
----

1. Within a module, create a Sub procedure called "a".

2. Within the same module, create a Sub named "A" and choose the OK button when the "duplicate definition" message is displayed.

The original procedure name is updated with the most recent Sub procedure name taking the place of the old Sub procedure name, despite the "duplicate definition" error message. To work around the problem, change the capitalization.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgOther

BUG: No Option Button Active (Dotted) in Frame

Article ID: Q76520

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you have more than one frame (or group) of option buttons, Visual Basic correctly marks one button as active (with a dot) in the first group of option buttons, but initially fails to place a dot anywhere in additional groups (or frames) of option buttons. You must manually click an option button in the additional group for the dot to appear in that group.

WORKAROUND
==========

If you want a particular option button selected (containing the dot) in a group or frame, set that button's Value property in the Form_Load event Procedure. For example:

```
   Option3.Value = -1
```

This will place a dot in the Option3 button in addition to the dot in the Option1 button when you run the program again.

Note that you cannot place a dot in both the Option1 button and the Option2 button if they are both placed in the same frame. By putting a group of options in one frame, you are specifying that the user may choose only one of the grouped options.

Note also that buttons on a form outside any frame behave as a group.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic with a new project.

2. Draw two frames on the form, with one frame being half the size of

   the form and the other frame being the other half of the form.

3. Place two option buttons in each of the frames by selecting the
   Option Button tool from the Toolbox and pointing, clicking, and
   dragging the option buttons onto the frames.

4. Run the program by pressing the F5 key. Note that there is only one
   option button containing a dot.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Italic and Large Fonts Display Poorly in Text Boxes
Article ID: Q76555

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Italic letters of any size are incorrectly truncated when typed in a text box. Also, if you use the BACKSPACE key to delete characters that are in italic text or large fonts, pieces of characters remain after the deletion.

CAUSE
=====

This problem is caused by Windows versions 3.0 and 3.1, not by Visual Basic.

WORKAROUND
==========

To work around this problem, you can use the Refresh method during the text box change event to correctly update the screen. However, this will also cause some visible flickering as you type characters into the text box.

To correct the appearance of the characters in the text box, add the following code to the text box's Change event.

```
   Sub Text1_Change ()
      Text1.Refresh
   End Sub
```

This code forces the text box to update the visual display every time time a change is made, so it corrects the problem but generates a flicker of the text box.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Windows versions 3.0 and 3.1. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Place a large text box on a blank form.

2. Set the text box FontSize property to any size above 12 points, or set the FontItalic property to True.

3. From the Run menu, choose Start.

4. Type anything in the text box.

5. Press the BACKSPACE key.

Note: If the font size is large, the font will be displayed correctly until the characters are removed with the BACKSPACE key. Italic characters will be displayed incorrectly when entered into the text box, and backspacing will truncate the deleted characters.

Additional reference words: buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Dir List Box Does Not Give Error 68 Device Unavailable
Article ID: Q76628

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

Under circumstances described below, error 68 (Device Unavailable) fails to display in conjunction with drive and directory list boxes. In the example given below, error 68 should display when drive A's door is open and the user clicks the directory list box.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start a New Project in Visual Basic. Form1 is created by default.

2. Add a drive list box and a directory list box to Form1.

3. Add the following code to the Sub Drive1_Change event procedure:

```
   Sub Drive1_Change ()
      On Error GoTo Trap
      Dir1.Path = Drive1.Drive
      Exit Sub
      Trap:
      Print Err
      Resume Next
   End Sub
```

4. Run the program by pressing the F5 key.

5. Select the down arrow of the drive list box by clicking the left mouse button. Select drive A. At this point, an error 68 should appear on the form.

6. Select the drive list box down arrow again. This time, select drive C.

7. Place a disk in drive A. Repeat step 5. No error message is displayed.

   The directory list box should be updated to display the A drive.

8. Open the drive A disk door. Then double-click in the directory list box.

Error 68 should be displayed, but isn't. Error 68, "Device Unavailable," should display when drive A's door is open and the user clicks the directory list box.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00 errmsg
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: FormName Not in Correct Order After Out of Memory Error
Article ID: Q76983

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If, when creating a new form, you receive an "Out of memory" error message, no form will be loaded. However, the default FormName is still incremented by 1 so that when a new form can be created (for example, by deleting an already existing form) after getting the error, the FormName of the newly-created form is not in the correct sequence.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

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

2. Create new forms by choosing New Form from the File menu (ALT, F, F) until you get an "Out of Memory" error. On one machine, this occurred when trying to load Form52. In Visual Basic version 2.0, this may not occur until you reach Form100 or more.

3. Choose the OK button to acknowledge the error message.

4. Delete Form51 (or whatever the final form is) by choosing Remove File from the File menu (ALT, F, R).

5. Create one more form. Form53 will be the next form, even though Form52 was never created.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes PrgOptMemMgt

BUG: LinkTimeOut of -1 Waits Only 6553.5 Secs Before Time Out

Article ID: Q77243

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Contrary to the documentation and online Help for Microsoft Visual Basic, setting the LinkTimeOut property of a control to -1 will not cause the control to wait forever for a DDE operation to complete. Setting the LinkTimeOut property to -1 will cause the control to wait for 65535 intervals of 1/10 second, for a total of approximately 1 hour and 49 minutes.

WORKAROUND
==========

To work around this problem, you can trap the DDE time-out error using the On Error statement in Visual Basic. If the error was "Timeout while waiting for DDE response," you can retry the DDE operation until it succeeds. The following is a code example:

```
   Sub DDE_Retry_Forever (Source as Control, commandx$)
      On Local Error Goto Handler

      Source.LinkExecute commandx$
      Exit Sub

      Handler:
      If Err = 286 Then
         Resume
      Else
         Error Err
      End If
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist1.00 buglist2.00 1.00 2.00
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: IAPDDE

BUG: DateSerial Does Not Give Error for Invalid Month or Day

Article ID: Q77393

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The DateSerial function doesn't generate an error when you use values for the month and the day arguments that are outside the ranges specified in the "Microsoft Visual Basic: Language Reference" version 1.0 manual.

You can use a numeric expression for each argument representing the number of days, months, or years before or after a certain date. But you will get an "Illegal function call" error message if you use a value for the year that is not between 1753 and 2078 (inclusive). You also get the error if the date specified by the three arguments either directly or indirectly evaluates to a date that is before January 1, 1753 or after December 31, 2078.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Page 65 of the "Microsoft Visual Basic: Language Reference" version 1.0 manual states the following:

"...the range of numbers for each DateSerial argument should conform to the accepted range of values for the unit. These values are 1 to 31 for days, and 1 through 12 for months. You can also specify relative dates for each argument by using numeric expressions representing the number of days, months, or years before or after a certain date...."

You can actually have values outside these ranges for the month and day argument and Visual Basic will not give an error. For example, a value of 0 for the day evaluates to the last day of the previous month. A value of 13 for the month translates to the first month (January) of the next year.

The following are examples of statements that will not produce errors:

```
   x# = DateSerial(63,7,12)    'evaluates to July 12, 1963
   x# = DateSerial(63,13,5)    'evaluates to January 5, 1964
   x# = DateSerial(63,7,33)     'evaluates to August 2, 1963
   x# = DateSerial(63,10,-1)    'evaluates to September 29, 1963
   x# = DateSerial(63,-1,5)     'evaluates to November 5, 1962
```

The following statements will generate an "Illegal function call" error because they produce dates before January 1, 1753 and after December 31, 2078:

```
   x# = DateSerial(1750,3,1)    'evaluates to March 1, 1750
   x# = DateSerial(2078,12,40) 'evaluates to January 9, 2079
   x# = DateSerial(1753,-5,20) 'evaluates to July 20, 1752
```

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: RefsDoc EnvtRun

BUG: Incorrect Focus Shift for Disabled Control in Break Mode
Article ID: Q77734

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

In Break mode in the Microsoft Visual Basic environment (VB.EXE), disabling a control or making a control invisible does not shift the focus to the next control in the tab order. Instead, the focus remains on the disabled control.

At run time, the focus correctly shifts to the next control in the tab order.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. From the File menu, choose New Project.

2. Place two controls on Form1 (in this example, two command buttons).

3. From the Run menu, choose Start.

4. From the Run menu, choose Break.

5. In the Immediate window, type the following:

```
      Command1.SetFocus
      Command1.Enabled = 0
      Print Screen.ActiveControl.Caption
```

   The active control will be Command1.

6. From the Run menu, choose Continue.

   Note: The disabled control, Command1, will still have the focus. To shift the focus to the next control in the tab index, press the TAB key.

If the example code from step 5 above is used at run time, the focus will correctly shift from Command1 to Command2.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Extra Click Event If Double-Click When Mouse Button Down

Article ID: Q77738

The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for Windows, versiona 2.0 and 3.0
 - Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When one mouse button is held down, double-clicking the other button generates one more Click event than necessary. The problem does not occur when double-clicking either mouse button individually.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The following code demonstrates that an extra Click event is generated when double-clicking one mouse button while holding the other down.

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic.

2. Double-click the form or press the F7 key to bring up the code window. Enter the following code in the Form_Click event procedure for Form1:

```
   Private Sub Form_Click ()
      Print "Click"
   End Sub
```

3. Enter the following code in the Form_DblClick event procedure:

```
   Private Sub Form_DblClick ()
      Print "DblClick"
   End Sub
```

4. Enter the following code in the Form_MouseDown event procedure:

```
   ' Enter the following two lines as one, single line:
   Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
      X As Single, Y As Single)
      Print "Down"; Button
   End Sub
```

5. Enter the following code in the Form_MouseUp event procedure:

```
' Enter the following two lines as one, single line:
Private Sub Form_MouseUp(Button As Integer, Shift As Integer,
   X As Single, Y As Single)
   Print "Up"; Button
End Sub
```

6. From the Run menu, choose Start.

7. Using the right mouse button, double-click anywhere on the form. The output to Form1 should be:

```
Down 2
Up 2
Click
DblClick
Up 2
```

8. Press and hold the left mouse button. The output to Form1 should be:

```
Down 1
```

9. While holding the left mouse button down, double-click with the right mouse button. The output to Form1 should be:

```
Down 2
Up 2
Click
DblClick
Up 2
Click
```

The last Click was not generated when double-clicking with the right mouse button alone (as illustrated in step 8 above). This additional call to the Click event procedure is not expected behavior and is a problem with Visual Basic. The problem also occurs when the right mouse button is held down and you double-click the left mouse button.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: CTRL+LEFT/RIGHT ARROW Behaves Differently When Edit/Type
Article ID: Q77928

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The key combinations CTRL+LEFT ARROW and CTRL+RIGHT ARROW work differently when editing code in a procedure than when typing in the Immediate window.

In the Immediate window, CTRL+LEFT ARROW will move the cursor in front of the preceding word even if that word is one of the following symbols:

```
   ! @ # $ % ^ ^ & * ( ) { } : ; , " ' [ ] < >
```

In the code editor, these symbols are not treated as words, so the cursor skips over them when using the ARROW key combinations to position the insertion point.

STATUS
======

Microsoft has confirmed this to be a bug in the VB.EXE environment of the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

In a code window, using the LEFT ARROW key with the CTRL key held down will move the cursor to the beginning of the preceding word or letter on that line, disregarding any punctuation marks and other symbols -- that is, any character obtained by typing a number while holding down the SHIFT key, all punctuation marks, brackets, braces, single quotation marks, and double quotation marks.

In the Immediate window, only the period is not treated as a word and is skipped over when using the CTRL+LEFT ARROW or CTRL+RIGHT ARROW key combination.

Steps to Reproduce Problem
----

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

2. Press F7 or double-click Form1 to bring up the code window.

3. Enter the following code in the Form_Click event procedure of Form1:

```
Sub Form_Click()
   print "Home."
End Sub
```

4. While the cursor is still at the end of the line, press CTRL+LEFT ARROW to move the cursor to the beginning of the previous word. The cursor should move directly in front of the H in Home.

5. From the Run menu, choose Start to run the program.

6. Press CTRL+BREAK to bring up the Immediate window.

7. Type the following in the Immediate window:

```
Print "Home."
```

9. With the cursor at the end of the line, press CTRL+LEFT ARROW. The insertion point should be directly in front of the last double quotation mark.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

FIX: Printing with HPPCL5A.DRV to HP LaserJet III Cuts Line

Article ID: Q78079

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Choosing Print from the Visual Basic File menu to print source code truncates one line of code per page of output when printing to a Hewlett-Packard (HP) LaserJet series III printer using the HPPCL5A.DRV printer driver.

CAUSE
=====

This is a problem with the Hewlett-Packard LaserJet series III printer driver version 3.42 for Windows.

STATUS
======

Microsoft has confirmed this to be a problem with the HPPCL5A printer driver version 3.42. This problem was corrected by the HP III driver version 30.3.85 included with Microsoft Word for Windows version 2.0.

Additional reference words: 1.00 2.00 HP laser jet truncate lose
KBCategory: kb3rdparty kbhw kbenv kbprg kbfixlist kbbuglist
KBSubcategory: EnvtDes

BUG: ToolBox Picture Control Bitmap Too Small on EGA
Article ID: Q78132

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The bitmap for the picture control (in the Toolbox window) in EGA mode
is 27 by 22 pixels, when it should be 28 by 22 pixels. The result is a
2-pixel thick black line at the left side of the picture control bitmap,
rather than a 1-pixel thick line it should be.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

Additional reference words: buglist1.00 buglist2.00 1.00 2.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Using Nonstandard Icons Can Cause UAE/GP Fault/Hang

Article ID: Q78380

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you create an icon in other than standard .ICO format and attach that icon to a Visual Basic form, you may get an Unrecoverable Application Error (UAE) in Windows version 3.0, a General Protection (GP) fault in Windows version 3.1, or your computer may hang (stop responding) and require you to turn the computer off to get out of it.

Icons in other than standard format might include a picture of how the icon looked when minimized or pasted directly to the form.

Nonstandard icons can also cause less severe run-time errors such as "Invalid Picture." The icon will load at design time but cause problems at run-time.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Icons created with utilities other than IconWorks -- even those created with the Windows Software Development Kit (SDK) Paint utility -- can cause problems because they may not conform to the standard .ICO format.

The standard .ICO format that Visual Basic supports is a 32 by 32 pixel matrix, which is specified in the icoDIBSize field in the header of the resource file. Because icons are handled as resources, once they are incorporated into the .EXE file, they can actually corrupt the code, which can cause the computer to hang during execution or cause a UAE or GP fault.

REFERENCES

"Microsoft Windows Software Development Kit: Reference Volume 2," version 3.0, page 9-2

"Microsoft Windows Programmer's Reference," Chapter 9, Microsoft Press, 1990

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbgraphic kbprg kbbuglist
KBSubcategory: APrgGrap

BUG: Multiline Text Box Contents Not Gray When Enabled=False
Article ID: Q78892

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
---------------------------------------------------------------------
```

SYMPTOMS
========

When the MultiLine property of a text box is True (-1) and the Enabled property is False (0), text inside the text box displays incorrectly as black instead of gray.

WORKAROUND
==========

To work around the problem, set the ForeColor property of the text box to gray when the Enabled property is set to False (-1) as shown in the example below.

```
'*** In the global module: ***
Global Const WINDOW_TEXT = &H80000008    ' from CONSTANT.TXT
Global Const GRAY_TEXT = &H80000011      ' from CONSTANT.TXT

' *** In the form: ***
' to disable a multi-line text box
text1.Enabled = 0                      ' disable
text1.ForeColor = GRAY_TEXT            ' gray

' to enable a multi-line text box
text1.Enabled = -1                     ' enable
text1.ForeColor = WINDOW_TEXT          ' black
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Visual Basic Code Window Hides Split View if Resized

Article ID: Q79057

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

The Visual Basic development environment (VB.EXE) behaves unexpectedly when a split Code window is resized. Instead of proportionally resizing the two sub-windows along with the parent window, the lower split view is obscured. The only indication that a split window is in effect is that both the horizontal scroll bar and the bottom of the vertical scroll bar are also obscured.

WORKAROUND

To work around the problem, resize the Code window to a convenient size before splitting the window.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Open a Code window.

2. Create a split Code window by placing the cursor between the Code window header and the top of the Code window and dragging downward.

3. Resize the Code window to a smaller size -- from the top down or from the bottom up.

The result is that the lower window is hidden, including any break points you were trying to track (for example, while watching one set in each window).

Additional reference words: buglist1.00 buglist2.00 1.00 2.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Invalid outside Sub Error When Copy or Paste to General

Article ID: Q79240

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

An "Invalid outside Sub or Function" error occurs in the VB.EXE environment under the following conditions:

- A Sub or Function is copied to the general Declarations section of a form.
- The name of the Sub or Function copied to the general Declarations section is changed, or the original Sub or Function that was copied is deleted.
- The program is run from the VB.EXE environment.

CAUSE
=====

The problem occurs when you copy a subprogram to the general Declarations section with Sub subname() and End Sub (or Function functionname () ... End Function) included. If you change the name of the original or copied Sub (or Function), the error "Invalid outside Sub or Function" will occur at run time. After the error occurs, the Sub or Function header of the copied Sub will be missing.

This problem occurs because the Sub or Function that was changed is treated as the entry of a new procedure. The body of the Sub or Function and the End Sub (or End Function) statement are treated as an existing part of the general Declarations section and are left behind.

The behavior is identical when the Sub (or Function) that was copied is deleted. The Sub (or Function) heading of the copy, residing in the general Declarations section, is treated as a new Sub or Function entry.

WORKAROUND
==========

Follow these steps to work around the problem:

1. Select (highlight) the remaining code fragment in the general Declarations section.

2. From the Edit menu, choose Cut (ALT, E, T).

3. From the Procedure box, choose Test2.

4. From the Edit menu, choose Paste to paste the code cut in step 2

above into the body of the Test2 subprogram.

5. Delete the duplicate End Sub statement.

Use the following steps to copy a subprogram and avoid the problem:

1. Create a new subprogram (such as Sub Test1).

2. Create a second subprogram with a different name (such as Sub Test2).

3. Copy just the body of the code from the first subprogram (Test1) into the second subprogram (Test2).

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. Enter the following code in the general Declarations of Form1:

```
Sub Test1 ()
   Print "Hello"
End Sub
```

3. Highlight the code for the Sub and press CTRL+INSERT to copy the entire Test1 subprogram.

4. Switch to the general Declarations window.

5. Paste the code copied in step 3 above by pressing SHIFT+INSERT.

6. Change the name of the Sub from Test1 to Test2.

7. From the Run menu, choose Start (ALT, R, S) to run the program.

The error occurs in the general Declarations section on the following code fragment:

```
      Print "Hello"
   End Sub
```

As illustrated above, the first line of the subprogram, Sub Test2 (), is missing. This is because Visual Basic treats the name change as a new Sub entry and established a new subprogram (Test2). The Procedure box will contain Test2 as a subprogram. Visual Basic considers the

remaining part of Test2 (the code fragment above) to be an existing
part of the general Declarations section.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Resetting ListIndex Property Generates Click Event
Article ID: Q79241

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

Resetting the ListIndex property of a list box, combo box, directory list box, or a file list box at run time generates a Click event for the control. For a drive list box, resetting the ListIndex property generates a Change event.

CAUSE

This is a result of the Windows subclass definition for these controls. When an item in one of these list boxes is selected, a Click event (or Change event for drive list box) occurs and the ListIndex property is updated. Conversely, when the ListIndex property is changed, a message occurs, generating the corresponding event.

WORKAROUND

Use the KeyUp procedure instead of click, and then call KeyUp when a key is pressed.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. Microsoft is researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

This behavior is not documented in the Visual Basic documentation or online Help. This behavior can cause some unexpected results. For example, if code in a Click (or Change) event procedure is assigning the selected items in the list box to an array (or directly to the Text property of another control), resetting the ListIndex property causes another such assignment, but with the new item.

If the ListIndex is reset to -1, a null item is assigned by the code because that setting indicates no item is selected.

Steps to Reproduce Problem

1. Start Visual Basic or from the File menu, choose New Project if
   Visual Basic is already running. Form1 is created by default.

2. Add a combo box (Combo1) to Form1.

3. Add a text box (Text1) to Form1.

4. Add a command button (Command1) to Form1.

5. Add the following code to the Click event for the list box chosen:

   ```
   Sub Combo1_Click ()
      text1.text = combo1.text
   End Sub
   ```

   Note that for drive and directory list boxes, change the assignment to:

   ```
      text1.text=drive1.list(drive1.ListIndex)

         -or-

      text1.text=dir1.list(dir1.ListIndex)
   ```

6. Add the following code to the Click event procedure for Command1:

   ```
   Sub Command1_Click ()
      combo1.ListIndex = -1
   End Sub
   ```

7. Add the following code to the Form_Load event procedure of Form1:

   ```
   Sub Form_Load ()
      For n = 1 To 10
         combo1.AddItem Format$(n, "0")
      Next
   End Sub
   ```

8. Run the program. Notice that when you click the Command1 button, the
   list box is updated as expected, the code in the Click event procedure
   for the list box is executed, and the Text property of the text box is
   changed.

REFERENCES
==========

"Programming Windows: the Microsoft Guide to Writing Applications for
Windows 3," by Charles Petzold, Microsoft Press, 1990

"Microsoft Windows Software Development Kit: Reference Volume 1,"
version 3.0

WINSDK.HLP file shipped with Microsoft Windows 3.0 Software
Development Kit

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00

KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Some Property Values May Be Incorrect in Maximized Form
Article ID: Q79242

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows versions 3.0 and 3.1

SYMPTOMS
========

The Top, Left, ScaleHeight, and ScaleWidth properties of a maximized Visual Basic for Windows form may return incorrect values. When a form is maximized, the values returned by these properties should be close to the resolution of your monitor. The only difference between the property values returned and the resolution should be due to BorderStyles, menus, or title bars, and should in no case be greater than the resolution of your monitor.

CAUSE
=====

In some cases, with a maximized form, the returned property values can be greater than the screen resolution. This is because of a problem in the Windows API routine, GetClientRect(), which Visual Basic calls to get the form properties. This is a problem with Microsoft Windows versions 3.0 and 3.1, not with Visual Basic.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Windows versions 3.0 and 3.1. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The Left property determines the distance between the internal left edge of an object and the left edge of its container. The Top property determines the distance between the internal top edge of an object and the top edge of its container. ScaleHeight sets or returns the range of the vertical axis for an object's internal coordinate system, and ScaleWidth sets or returns the horizontal axis. On a form, the coordinate system includes the form's internal area, not including borders and title bar.

Steps to Reproduce Problem

To duplicate the problem, experiment with various BorderStyles, set ScaleMode to pixels, and add the following code:

```
   Sub Form_Click()
```

```
      Print Left,Top,ScaleWidth,ScaleHeight
   End Sub
```

Run the application and click the form. Note the values printed. With no border, the values should correspond to the resolution of your monitor, and should change slightly for each BorderStyle from the addition of borders, menus, and title bars.

Here's another example. This occurs when you use the following code in a maximized form with a ScaleMode of 1 (twips) in a 800-by-600 (pixel) screen resolution:

```
   Sub Form_Click
      Print "Screen = "; screen.width; ", "; screen.height
      ' Enter each Print statements on one, single line.
      Print "Form = "; form1.width; ", "; form1.height;
                    " at "; form1.left; ", "; form1.top
      Print "-------------------------------------------------------
             -------------------------"
   End Sub
```

The following is the results:

```
   Screen = 12000, 9000
   Form = 12120, 9120 at -60, -60
```

Additional reference words: buglist3.00 buglist3.10 1.00 2.00 3.00 3.10
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Option Button w/ Focus Selected When Click Form Caption
Article ID: Q79602

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

In Visual Basic, if you run a program that contains an option button group, and one of the option buttons is not selected but has the focus, that option button will be selected -- causing the option button Click event -- when you select the form title bar, Minimize button, Maximize button, control menu box, or form size handles. This is does not occur with other Windows programs.

An option button Click event will also occur incorrectly on a form Load event if the option button is the only control on the form or if the option button's TabIndex property is set to 0. When the TabIndex property is 0, the option button is the control that gets the focus, causing a Click event for the option button. Putting another control on the form and setting that control's TabIndex to 0 solves the problem.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem Described in First Paragraph Above

1. Start 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 two option buttons to Form1.

3. From the Run menu, choose Start (ALT, R, S) to run the program.

4. Give the unselected option button the focus. This can be done by clicking the unselected option button and holding down the mouse button until you have moved the mouse cursor off of the form completely.

5. Click the form's title bar, control menu box, Minimize/Maximize button, or the resize handles. This will result in the option button being selected.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00

3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: SendKeys Causes Erratic Mouse Behavior on IBM PS/2

Article ID: Q79603

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows version 3.0

SYMPTOMS
========

When a Visual Basic program executes the SendKeys statement on an IBM PS/2 computer, Windows behaves erratically when you move the mouse until it is shut down.

CAUSE
=====

The erratic behavior is caused by continuous phantom mouse clicks and mouse movements.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Windows version 3.0. This bug was corrected in Microsoft Windows version 3.1.

MORE INFORMATION
================

If you are running Microsoft Windows 3.0 on a PS/2 computer and you press the NUM LOCK key while moving the mouse, the mouse events become erratic. The Visual Basic SendKeys statement affects the NUM LOCK key, so this problem results -- just as if NUM LOCK were pressed.

When you move the mouse, phantom Click events result in symptoms such as applications unexpectedly launching, or the mouse pointer jumping around the screen.

This problem has been reported to happen on the IBM PS/2 Model 50, Model 50z, Model 60, and Model 80.

Additional reference words: noupd 1.00 3.00 3.10 NUMLOCK
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Click Event May Fail to Occur in Cascading Menu

Article ID: Q80023

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

There is an inconsistency with the Click events of cascading menus in
Visual Basic. This problem occurs when hidden menus are displayed.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

If you design menus with cascading menus, you can process the Click
event for the menu selection that cascades another submenu. Conversely,
if you initially design the menu so that the menu Visible property is set
to False, you will not always be able to process the Click event for that
menu selection that cascades another menu.

Steps to Reproduce Problem
----

1. From the menu design dialog box of Visual Basic (VB.EXE), create
   a set of menus using the following table as a guide:

Caption	CtlName (or Name)	Level	Visible
A	MID_A	1	True
1	MID_ONE	2	
Cascade 1	CASCADE1	3	
B	MID_B	1	False
2	MID_TWO	2	
Cascade 2	CASCADE2	3	

2. Add two command buttons (Command1 and Command2) to the form.

3. Add the following code to your program in the appropriate places:

```
   Sub Command1_Click ()
      MID_A.Visible = -1
      MID_B.Visible = 0
   End Sub
   Sub Command2_Click ()
      MID_A.Visible = 0
      MID_B.Visible = -1
   End Sub

   Sub MID_TWO_Click ()
      Print "Cascade 2"
   End Sub

   Sub MID_ONE_Click ()
      Print "Cascade 1"
   End Sub
```

4. Run the program.

5. Click the A menu, then click the 1 menu. Notice that "Cascade 1" is printed to the form. Note that you may have to do this twice because the menu overlaps the display and erases most of it the first time.

6. Click the Command2 button to hide the A menu and show the B menu. Click the B menu, then click the 2 menu. Notice "Cascade 2" does not print to the screen as it did in step 5 above.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: TAB Character Can Incorrectly Cause KeyUp/KeyDown Events

Article ID: Q80286

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

Under certain circumstances, the TAB key may generate either or both a KeyDown or KeyUp event for a form or control. The Language Reference for Visual Basic version 1.0 states on page 160 that KeyDown and KeyUp events are not generated for the TAB key. This is normally true.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

The TAB key is normally used to switch focus from one control to another in the predefined tab order. This action does not normally produce a KeyDown or KeyUp event. However, if there is not another control that can accept the focus, pressing TAB generates a KeyUp and/or KeyDown event. This problem manifests itself in several situations:

- A form with no controls
- A form with only one control
- A form with all controls disabled (or all except one)
- A form with all controls invisible (or all except one)
- A combination of the last two above

Steps to Reproduce Problem

1. Start Visual Basic, or if it is already running, choose New Project from the File menu. Form1 is created by default.

2. Draw a command button on Form1.

3. Add the following code to the KeyDown event for the command button:

   ```
   Form1.Print KeyCode
   ```

4. Run the program.

5. Press the TAB key. The character 9 will appear on the form. The

   character 9 is the ANSI code for the TAB character.

6. End the program.

The TAB key should never produce a KeyDown or KeyUp event. However,
because this is a problem that may be corrected in future versions,
you should not write code that depends upon this behavior.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: No Resources Causes Failed to Open Graphics Server Error
Article ID: Q80780

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

With the Visual Basic programming environment (VB.EXE) running and with very low available Windows resources (0 to 1 percent), attempting to load the GRAPH.VBX Visual Basic custom control generates these misleading messages:

```
   Failed to open Graphics Server.
   GSW.EXE must be available via the DOS path.
```

followed by another error message:

```
   Can't load the custom control DLL: "C:\VB\GRAPH.VBX"
```

RESOLUTION

These messages incorrectly imply that the problem is that GSW.EXE is not in the MS-DOS path, when in fact the custom control could not load because of a lack of Windows resources (memory).

STATUS

Microsoft has confirmed this to be a bug in the GRAPH.VBX custom control provided with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

A prerequisite to re-create this problem is to deplete Windows system resources until the Program Manager Help About dialog box reports 1 percent or less resources available. To verify the level of resources available, from the Program Manager Help menu, choose About.

One way to deplete Windows resources is to launch as many sessions of NOTEPAD.EXE as possible before getting an error message (start Visual Basic before all of the Notepad sessions).

Steps to Reproduce Problem

With 1 percent or less resources available, the following procedure
will generate the above error messages:

1. From the File menu, choose New Project. Form1 is created by default.

2. From the File menu, choose Add File, and select the GRAPH.VBX
   custom control.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Gauge Custom Control: No Error for Illegal NeedleWidth
Article ID: Q80905

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

When you use the Gauge custom control, setting the NeedleWidth property to an invalid value fails to generate an error. Furthermore, attempting to set the NeedleWidth property outside its valid range will reset the NeedleWidth property to 1. This behavior occurs at both design time and run time.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File (CTRL+F12), and select GAUGE.VBX to add the Gauge control to the Toolbox. The Gauge tool will appear in the Toolbox.

3. Add the Gauge control to Form1 and set the gauge's Style property to 2 - 'Semi' Needle or 3 - 'Full' Needle.

4. Add the following code to the Form_Click event procedure.

```
   Sub Form_Click ()
      Gauge1.NeedleWidth = -3
      MsgBox "NeedleWidth = " + Str$(Gauge1.NeedleWidth)
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) to run the program.

Notice that clicking the form produces a message box that displays the value of the gauge's NeedleWidth property. Even though the NeedleWidth is explicitly set to -3 before the message box is displayed, the NeedleWidth property resets to a value of 1.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Grid Custom Control: Scroll Bars Displayed Unnecessarily
Article ID: Q80967

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

Under the following conditions, the Grid custom control incorrectly displays horizontal and vertical scroll bars when all the columns and rows fit in the control (which eliminates the need for scroll bars):

- The ScrollBars property is set to 3 (Both).
- The distance between the right column and the right edge of the control is less than the default width of a column.
- The distance between the bottom row and the bottom edge of the control is less than the default width a row.

WORKAROUND
==========

To work around this problem, add the following statements to the Form_Load procedure to set the ScrollBars property to 0 (none), then back to the original setting.

```
   Sub Form_Load ()
      save% = Grid1.ScrollBars    ' save setting
      Grid1.ScrollBars = 0        ' turn off scroll bars
      Grid1.ScrollBars = save%    ' restore setting
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the GRID.VBX custom control file. The Grid tool appears in the Toolbox.

3. Place a grid named Grid1 on Form1.

4. Set the grid properties Cols and Rows each to 3.

5. Size the grid so that all columns and rows are visible. Leave a small space between the grid area and the edge of the control.

6. From the Run menu, choose Start, or press F5 to run the program. Both horizontal and vertical scroll bars incorrectly appear.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Gauge Custom Control: Valid NeedleWidth Range 1 to 32767

Article ID: Q81187

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When you add the Gauge custom control control to a form, the NeedleWidth property incorrectly displays a value of 0 in the Settings box on the Properties bar. After running the Visual Basic application, the Settings box will display the correct default value of 1, unless the property was modified during run time.

RESOLUTION
==========

The valid range for the NeedleWidth property of the Gauge custom control is 1 to 32,767. Attempting to set the NeedleWidth property to a value outside this range resets the value to 1.

STATUS
======

Microsoft has confirmed this to be a bug in the Gauge custom control provided with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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 GAUGE.VBX file to your project by choosing Add File (CTRL+F12) from the File menu and selecting GAUGE.VBX from the appropriate directory.

3. Add the Gauge control to Form1.

4. Select the NeedleWidth property from the Properties list box to display the default NeedleWidth value. Note that the value is set to 0, which is outside the valid range of this property.

5. From the Run menu, choose Start (ALT, R, S) to run the program.

6. Double-click the form's Control box to end the application.

7. Again, select the NeedleWidth property from the Properties list box
   to display the default NeedleWidth Value. Note that the value is
   now set to 1.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: 3-D Panel Control Doesn't Resize to Key Status Control

Article ID: Q81449

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Unpredictable results occur if a 3-D Panel custom control's AutoSize property is set to 3 (AutoSize Child To Panel) and you place a Key Status control with its AutoResize set to True on the 3-D Panel as a child control. For example, the Key Status control keep may keep its default size and move to the upper left corner of the panel, and the Key Status control's top and left sizing handles may flash.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

If a single control is placed on a 3-D Panel custom control with AutoSize = 3 (AutoSize Child To Panel), the child control resizes to exactly fit within the panel's inner bevel. This setting for the AutoSize property has no effect if there are no child controls, more than one control, or if the panel has no bevels. If the child control has a fixed dimension (for example, the height of a combo box or a drive box), then that dimension of the panel will be adjusted to fit the child control instead while the other dimensions are resized to fit the panel.

The Key Status custom control has its AutoSize property default set to True, so its dimensions cannot be changed. However, 3-D Panel does not resize to the size of the Key Status control. Instead, when you draw a Key Status control onto a 3-D Panel with AutoSize = 3 and release the mouse button, the Key Status control keeps its default size and moves to the upper left corner of the panel, and the Key Status control's top and left sizing handles will flash. Also, the sizing handles of the Key Status control that you initially draw remain on the panel. Notice that the size of the control in the right box on the Properties bar will alternate between the size of the 3-D Panel and the size of the Key Status control.

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the THREED.VBX custom control file. The 3-D Panel tool will appear in the toolbox.

3. Place a 3-D Panel control (Panel3D1) on Form1.

4. Set its AutoSize property to 3 (AutoSize Child To Panel).

5. Draw a Key Status control on the panel.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Vertical Linear Gauge Loses Upper Border's Bottom Pixels
Article ID: Q81460

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

The fill area defined by the InnerXXX properties of the Gauge custom control overwrites the bottom-most line of pixels in the top border as defined by the InnerTop property. This behavior occurs only with the vertical linear style gauge.

STATUS
======

Microsoft has confirmed this to be a bug in the Gauge custom control provided with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File (CTRL+F12). In the Files box, select the GAUGE.VBX custom control file. The Gauge tool will appear in the toolbox.

3. Add the Gauge control to Form1, and set its properties to the following:

Property	Value
BackColor	&H00000000&
ForeColor	&H000000FF&
InnerTop	1
Style	1 - Vertical Bar

4. Add the following code to the Gauge_Click event procedure. (Make sure you add this code to the Click event procedure, not the Change event.

```
   Sub Gauge_Click ()
      Gauge1.Value = Gauge1.Max
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) to run the program.

When you click the Gauge, the top border of the Gauge will disappear.

Note: If you assign the Picture property to a bitmap and change the gauge's Value property to greater than 0, the bottom-most line of pixels in the top border will be redrawn in the Background color.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: InnerBottom/InnerRight Defines Gauge Fill Area Badly
Article ID: Q81461

```
---------------------------------------------------------------------
The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0
---------------------------------------------------------------------

SYMPTOMS
========

When you add the Gauge custom control to a form, the fill area defined
by the InnerXXX properties is incorrect. Specifically, the InnerBottom
sets the bottom border (InnerBottom - 1) pixels from the bottom-most
position of the control. Similarly, the InnerRight property sets the
right border (InnerRight - 1) pixels from the rightmost position of
the control. This behavior occurs only in the InnerBottom and
InnerRight properties.

WORKAROUND
==========

To work around the problem, set InnerRight to (InnerLeft - 1) and
InnerBottom to (InnerTop - 1) to create symmetrical borders. Note that
in order to create a border of set width, you must account for the
aspect ratio of your video display.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

Steps to Reproduce Problem
--------------------------

1. Start 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. From the File menu, choose Add File (CTRL+F12). In the Files box,
   select the GAUGE.VBX custom control file. The Gauge tool will
   appear in the toolbox.

3. Add the Gauge control to Form1 and set the gauge's properties as
   follows:

   Properties    Values
   -------------------------------
   InnerBottom   1
   InnerLeft     1
   InnerRight    1
   InnerTop      1
   ForeColor     &H000000FF& (Red)
   Value         100
```

Notice that the bottom and right borders have completely disappeared. This problem can also be illustrated by setting BackColor and ForeColor to different colors. When InnerLeft is equal to InnerRight, the left and right borders are not symmetrical. The same holds true for the InnerTop and InnerBottom properties.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Graph: ExtraData May Not Say: Invalid Property Value
Article ID: Q81472

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When you use the Graph custom control, the ExtraData property will not always generate an "Invalid Property Value" error when you assign it invalid numbers.

CAUSE
=====

ExtraData has different valid ranges, depending on which GraphType you are using. The widest range is from 0 to 15, inclusive. Even if values between 0 and 15 are not within the documented range for an individual GraphType, they may not generate an error.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

If you are using the 3-D bar graph (GraphType = 4), the ExtraData property holds the color values for the sides of the bars. Color values range from 0 to 15, so the legal values for ExtraData also range from 0 to 15. If you are using the 2-D pie graph (GraphType = 1) or the 3-D pie graph (GraphType = 2), the value of ExtraData will determine whether or not a pie piece is exploded from the graph. The documented range for ExtraData with pie graphs is from 0 to 1, where 0 = False and 1 = True. In practice, however, the range for ExtraData with pie graphs is from 0 to 15, where even values equal False and odd values equal True.

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the GRAPH.VBX custom control file. The Graph tool will appear in the

toolbox.

3. Add a graph control (Graph1) to Form1.

4. In the Properties list box, set the following values and properties:

Property	Value
GraphType	2
DrawMode	2
NumSets	1
NumPoints	5
ExtraData	0, 1, 14, 15, 16

As you assign the values for ExtraData, you will see:

- No change when ExtraData is set to 0.
- The second data point will be exploded when ExtraData is set to 1.
- No change when ExtraData is set to 14 (even numbers less than 16 = FALSE).
- The fourth data point will be exploded when ExtraData is set to 15 (odd numbers less than 16 = TRUE).
- An "Invalid Property Value" message generated when ExtraData is set to 16.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Graph Custom Control Text Disappears in EGA Video Mode
Article ID: Q81949

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions
  2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

When using the Visual Basic Graph custom control in an EGA video mode
with the Graph control Background property value set to dark gray and
the Foreground property value set to light gray, the text on the graph
will disappear.

CAUSE
=====

This is a known problem with Windows versions 3.0 and 3.1. This is not
a problem with the Graph custom control or with Visual Basic.

STATUS
======

Microsoft has confirmed this to be a problem with Microsoft Windows
versions 3.0 and 3.1. We are researching this problem and will post
new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

Windows defines dark gray as the color created when red=128, blue=128, and
green=128. Windows defines light gray as the color created when red=192,
blue=192, and green=192.

Windows, when given light gray text on a dark gray background in EGA video
mode, alters the value of the text color to dark gray, which is the closest
representation it can make in that video mode. The subsequent dark gray
text on a dark gray background makes it appear as though the text has
disappeared.

The Visual Basic Graph custom control allows you to set the background
and foreground colors to 16 predefined colors. Colors 7 and 8 are
light gray and dark gray, respectively. Graph uses Windows values for
dark gray and light gray, and so displays the same video problems as
Windows itself.

Steps to Reproduce Problem
----

1. Set the video mode of Windows to EGA.

2. Re-enter Windows if necessary and start Visual Basic.

3. In the Visual Basic environment with the VB Graph custom control
   loaded, create a form (Form1).

4. Add a Graph custom control (Graph1).

5. Set Graph1.DrawMode=2 (draw).

6. Set Graph1.Background=8 (dark gray) and Graph1.Foreground=7 (light
   gray).

The text disappears, leaving colored bars on a dark gray background.

Additional reference words: 1.00 2.00 3.00 buglist3.00 buglist3.10
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Scroll Control: UAE/GPF If Drag Method in GotFocus Event
Article ID: Q81955

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

An Unrecoverable Application Error (UAE) in Windows version 3.0 or a General Protection (GP) fault at address 0012:001E in Windows version 3.1 may occur when you perform a manual drag (using the Drag method) in the GotFocus event for an Instant Change Scroll Bar custom control (INSTSCRL.VBX in Visual Basic version 1.0) or the regular Scroll Bars control (in Visual Basic version 2.0 or 3.0) and change the focus to the Scroll Bar (or the Instant Change Scroll Bar custom control) in either the Change or the Changing event.

The Instant Change Scroll Bar custom control comes with the Microsoft Visual Basic Professional Toolkit version 1.0 for Windows. The capabilities of the Instant Change Scroll Bar were implemented in the regular Scroll Bars controls in both the Standard and Professional editions of Visual Basic versions 2.0 and 3.0.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem in Visual Basic Version 1.0
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the INSTSCRL.VBX custom control file. The Instant Change Scroll Bar tools appear in the toolbox.

3. Place an InstHScroll or InstVScroll control and a command button on Form1.

4. Double-click the Instant Change Scroll Bar control (or press F7) to open the Code window. Enter the following code in the Changing event:

```
   Sub InstHScroll1_Changing ( )
      Command1.TabIndex = 0
      InstHScroll1.TabIndex = 1
   End Sub
```

   Add the following code in the GotFocus event:

```
   Sub InstHScroll1_GotFocus ( )
      InstHScroll1.Drag 1
   End Sub
```

5. Press F5 to run the example. Click the scroll arrow of the Instant Change Scroll Bar and wait a few seconds. A UAE or GP fault will occur.

This problem occurs with both the InstVScroll and InstHScroll controls, and with the code above in either the Change or Changing events.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Grid: No Error Changing FixedAlignment on Non-Fixed Col

Article ID: Q81998

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Using the Grid custom control, setting the text alignment of a non-fixed column with the FixedAlignment property will not generate an error. Though this value is saved, it does not affect the text alignment of the specified non-fixed column.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Behavior
----

1. Start 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. From the File menu, choose Add File (CTRL+F12), and select GRID.VBX to add the Grid control to the Toolbox. The Grid tool appears in the Toolbox.

3. Add a Grid control to Form1.

4. Add the following code to the Grid_Click event procedure:

```
   Sub Grid_Click ()
      Grid1.ColWidth(1)=2000
      Grid1.Col=1
      Grid1.Row=1
      Grid1.Text="Hello"
      Grid1.FixedAlignment(1)=1  'Right Justify
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) to run the program.

Notice that when you click the grid, the FixedAlignment property accepts the new value, but the alignment of the text does not change.

Note that if you try to do the opposite (that is, attempt to set the
text alignment of a fixed-column with the ColAlignment property), an
"Invalid Column" error message will be displayed.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Graph: AutoInc Increments ThisPoint Instead of ThisSet
Article ID: Q81999

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The Graph custom control version 1.2 has four array properties: ColorData, LegendText, PatternData, and SymbolData. The values of these properties directly affect sets of data rather than the individual points in the sets. With the AutoInc property set to True, assigning a value to these four arrays will increment ThisPoint rather than ThisSet. This behavior is a potential cause of logic errors in code.

WORKAROUND
==========

To work around the potential logic problems caused by incrementing ThisPoint, you should occasionally reset the AutoInc incrementing position by assigning values for ThisSet and ThisPoint in your code.

A second workaround is to set AutoInc to False (AutoInc=0), and explicitly set ThisSet and ThisPoint before entering a piece of data.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

There are eight array properties in Graph: GraphData, ExtraData, LabelText, XPosData, ColorData, LegendText, PatternData, and SymbolData. To access an individual point in these arrays, you need to set the ThisSet and ThisPoint properties to indicate that point. If AutoInc is set to True (AutoInc=1), Graph will automatically set the ThisPoint and ThisSet properties.

AutoInc increments ThisSet and ThisPoint differently, depending on which property is being accessed. AutoInc will increment both ThisSet and ThisPoint when adding data to the GraphData property. For all other array properties (ExtraData, LabelText, XPosData, ColorData, LegendText, PatternData, and SymbolData), AutoInc will only increment ThisPoint. The data that you assign to the ExtraData, LabelText, and XPosData apply to the individual points of a set, so logically AutoInc should only increment ThisPoint. However, the data that you assign to the ColorData, LegendText, PatternData, and SymbolData array properties apply to the separate sets. In these cases, AutoInc should logically be incrementing the ThisSet property, but in practice it increments only the ThisPoint property.

Note: AutoInc is incrementing the proper values internally, so the data assigned to these four array properties is accurate and will function properly. AutoInc displays its progress by also incrementing ThisPoint, which is not always the logical choice.

The following example demonstrates how AutoInc increments ThisPoint and ThisSet when assigning values to ColorData. To test another array property, substitute that array name for ColorData.

Steps to Reproduce Problem

1. With Visual Basic running and the Graph custom control loaded, create a form (Form1).

2. On Form1, add a command button (Command1), a picture box (Picture1), and a graph control (Graph1).

3. Change the following properties for Command1:

Control	Property	Value
Command1	Caption	"Start"
Graph1	AutoInc	1 (true)
Graph1	NumSets	2
Graph1	NumPoints	3

4. Add the following code to the Command1 Click event:

```
Sub Command1_Click ()

   Command1.Caption = "ColorData"   'set caption equal to array
                                    'property name to be tested
   Picture1.Cls
   Picture1.Print "ThisSet", "ThisPoint"
   ' loop through full array:
   For i = 1 To Graph1.NumSets * Graph1.NumPoints
      Picture1.Print Graph1.ThisSet, Graph1.ThisPoint
      Graph1.ColorData = 1  'Make some valid assignment so
                            ' AutoInc increments
   Next
   Graph1.DrawMode = 2              'display newly assigned values

End Sub
```

5. Press F5 to run the program.

When you run the program and click the Command1 button, the program will display the array property being tested, and the picture box will display the increment pattern of ThisSet and ThisPoint as the program loops through the array property. The graph is then updated to display the newly assigned values.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00 buglist1.00 buglist2.00
buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Animated Button: 8 Pt. Roman/Mdrn Fonts Don't Underline

Article ID: Q82004

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

Small (8 point) Roman and Modern fonts will not underline on EGA systems when using the Animated Button custom control. (ANIBUTTON.VBX.)

STATUS

Microsoft has confirmed this to be a bug in the Microsoft Professional Toolkit for Microsoft Visual Basic programming system version 1.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

On an EGA system:

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. From the File menu, choose Add File. In the Files box, select the ANIBUTON.VBX custom control file. The Animated Button tool appears in the toolbox.

3. Create a default Animated Button control by double-clicking the animated tool in the toolbox.

4. Set the following properties from the Properties Bar:

```
      FontName = Modern (or Roman)
      FontSize = 8
      FontUnderline = True
```

Notice that the caption is not underlined as it is on a VGA system. If the FontSize is changed to a larger size, the underline will appear. The underline will also appear on fonts other than Roman or Modern.

Additional reference words: buglist1.00 1.00
KBCategory: kbgraphic kbprg kbbuglist
KBSubcategory: APrgGrap

BUG: Graph Axis Titles Don't Switch on Horizontal Bar Graphs
Article ID: Q83463

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

The Graph custom control allows you to convert your graph control from a non-horizontal bar graph to a horizontal bar graph or vice versa. This conversion will switch all necessary information to its proper position except for the axis titles. The BottomTitle and LeftTitle should switch positions, but do not.

WORKAROUND
==========

As a workaround for this problem, test whether the graph is being converted from or to a horizontal bar graph, and switch the values for BottomTitle and LeftTitle yourself. The example shown in the More Information section illustrates the problem and provides code to work around it.

STATUS
======

Microsoft has confirmed this to be a bug with the Graph custom control supplied with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Step-by-Step Example
----

1. Start 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. From the File menu, choose Add File. In the Files list box, select the GRAPH.VBX custom control file. The Graph tool appears in the toolbox.

3. On Form1, add three command buttons (Command1, Command2, and Command3) and a Graph control (Graph1).

4. Change the following properties:

```
   Control     Property       Value
   -----------------------------------------------------
```

Command1	Caption	Make Horizontal
Command2	Caption	Make Vertical
Command3	Caption	Switch Correctly
Graph1	Width	4000
Graph1	Height	2500
Graph1	GraphType	3 (default)
Graph1	GraphStyle	0 (default)
Graph1	GraphData	10, 20, 30, 40, 50
Graph1	BottomTitle	Title for axis labeled 1-5
Graph1	LeftTitle	Title for axis labeled 0-50

5. In the Command1 Click event, add the following code:

```
Sub Command1_Click ()
   Graph1.GraphStyle = 1    'horizontal
   Graph1.DrawMode = 2     ' redraws graph with new properties
End Sub
```

6. In the Command2 Click event, add the following code:

```
Sub Command2_Click ()
   Graph1.GraphStyle = 0 'default (vertical)
   Graph1.DrawMode = 2   ' redraws graph with new properties
End Sub
```

7. In the Command3 Click event, add the following code:

```
Sub Command3_Click ()
   Const TRUE = 1
   OldStyle = Graph1.GraphStyle
   Graph1.GraphType = 3  'or change according to your needs
   Graph1.GraphStyle = 1  'or change according to your needs

   If (Graph1.GraphType=3) Or (Graph1.GraphType=4) Then BarGraph%=TRUE
   ' The next line of code takes advantage of the fact that the
   ' GraphStyle numbers for the horizontal bar graphs are odd and the
   ' vertical are even.
   If (Graph1.GraphStyle + OldStyle) Mod 2 = 1 Then Switched% = TRUE

   If BarGraph% And Switched% Then
      temp$ = Graph1.BottomTitle
      Graph1.BottomTitle = Graph1.LeftTitle
      Graph1.LeftTitle = temp$
   End If
   Graph1.DrawMode = 2
End Sub
```

8. Press F5 (or ALT, R, S) to run the program.

When you run the program and click the Command1 button, Graph1 will redraw itself as a horizontal graph. The left and bottom labels switched but the LeftTitle and BottomTitle do not. Next, click the Command2 button. The Command2 Click event will return the graph to its original appearance.

When you click the Command3 button, Graph1 will redraw itself as a horizontal graph with all labels and titles switched appropriately.

The code for the Command3 Click event was written to react appropriately, regardless of which GraphType and GraphStyle are chosen.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Omitting Year for DateValue May Give Unexpected Results

Article ID: Q84115

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you omit the year portion of the DateValue function argument, DateValue uses the current year from the computer's system date. However, if you also pass an invalid day for the month, DateValue interprets the month as the year and the day will default to 1. For example, 3/30 will be interpreted as 3/30/92, but 3/44 will not produce an error message, and will be interpreted as 3/1/44.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The DateValue function returns a serial number that represents the date of the string argument. The date string can be in various forms. For example:

```
   3/30/92
   3/30/1992
   March 30, 1992
   Mar. 30, 1992
   30-Mar-1992
   30 March 92
```

The year portion of the string argument may be omitted, in which case the current year of the computer's system date is used. For example, 3/30 will cause DateValue to return the serial number that represents 3/30/92 (if 1992 is the year of the system date).

However, if the year is omitted and the day is not a valid day for that month of the current year, the month will be interpreted as the year and the day will default to 1. So 3/44 will be interpreted as 3/1/44.

Steps to Reproduce Problem
----

1. Start 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 code to the Form_Load event procedure:

```
   Sub Form_Load ()
      Debug.Print "3/30 ="; DateValue("3/30")
      Debug.Print "3/30/92 ="; DateValue("3/30/92")
      Debug.Print
      Debug.Print "3/44 ="; DateValue("3/44")
      Debug.Print "3/1/44 ="; DateValue("3/1/44")
   End Sub
```

3. Press F5 to run the program.

Notice in the Immediate window that the serial numbers returned by the DateValue function for 3/30 and 3/30/92 are the same (assuming 1992 is the current year of the system date), and the serial numbers for 3/44 and 3/1/44 are the same. Also, no error message was produced.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgOther

BUG: VB Graph Custom Control: SeeThru Paints Incorrectly
Article ID: Q84236

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS

When you use the Graph custom control with the SeeThru property set to True, Graph fails to paint properly. The Graph custom control will not repaint itself to show a see-through background nor to show updated information. Often it will create obvious holes through its parent form.

In addition, if anything on the form is under the Graph custom control, the overlapped region won't print when you execute PrintForm even though you see it on top when you print. This occurs most often when you have two overlapped Graph controls -- one with SeeThrough set to True, the other with SeeThru set to False.

STATUS

Microsoft has confirmed this to be a bug in the Graph custom control supplied with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

The Graph SeeThru property is supposed to have a clear background when it is set to True. This property allows any text or bitmaps displayed on the parent form to show through. However, the SeeThru property does not actually behave this way.

When the SeeThru property is first set to True and the graph is repainted by setting DrawMode = 2, the background color does not become clear. Also, the graph is not repainted, but rather just painted again on top of itself. If any other properties or data were reset before the DrawMode = 2 call is made, the changes might overlap the old settings, or not appear at all.

If circumstances call for the Graph control to completely repaint itself (such as when the parent form is minimized and then maximized), Graph will not repaint at all. Because Windows is expecting Graph to paint that region, it will not repaint the parent form behind the control. Graph also does not paint that region, so a hole is left in the form that shows the desktop behind the parent form. If you try to force Graph to repaint itself by setting DrawMode = 2, the actual Graph (without the background) will appear in the hole on top of the desktop clutter.

Steps to Reproduce Problem

1. Start 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. From the File menu, choose Add File. In the Files box, select the GRAPH.VBX custom control file. The Graph tool appears in the Toolbox.

3. On Form1 add a graph control (Graph1), and two command buttons, Command1 and Command2.

4. Change the following properties for the Command buttons:

Control	Property	Value
Command1	Caption	SeeThru
Command2	Caption	DrawMode

5. Add the following code to the Command1_Click event:

   ```
   Sub Command1_Click ()
      Graph1.SeeThru = 1
   End Sub
   ```

6. Add the following code to the Command2_Click event:

   ```
   Sub Command2_Click ()
      Graph1.DrawMode = 2
   End Sub
   ```

7. Press F5 to run the program.

   When you run the program, Graph1 appears normal. If you click the Command2 button to repaint Graph1, you will see the old graph being erased and then replaced by a new version of it. Because the random data generator inherent to Graph was left on, new data will be displayed. This is normal behavior. If you minimize and then maximize Form1, Graph1 will repaint itself correctly.

8. Click Command1 to turn on the SeeThru property, and then click on Command2 to repaint Graph1.

   This time Graph1 does not disappear before being redrawn. Instead, the new version of the graph is just painted on top of it and the background color is still there. Again, because the random data generator was left on, new data should be displayed. If the new data values are less than the old values, they won't be seen. The bars on Graph1 will appear to continuously rise every time the Command2_Click event is triggered.

9. Minimize Form1.

   Look at the area of the desktop where the graph control used to be. You will notice that it remains after the form is maximized.

10. Maximize Form1.

    The desktop still appears where the Graph1 control should be. If you click the Command2 button, the graph alone will be printed in the rectangle where Graph1 should be. Again, the graph will paint on top of itself instead of repaint itself every time you trigger the Command2_Click event.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Must Call API to Print Color Text on Color Printer in VB
Article ID: Q84269

The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for
   Windows, versions 2.0 and 3.0
 - Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Visual Basic for Windows does not directly support printing text in color to a color printer.

WORKAROUND
==========

To print in color, you must first make a call to the Windows API function SetTextColor(). The example below shows how to implement this call into a Visual Basic application to allow for printing of colored text.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post more information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The ForeColor property of the Printer object was not fully implemented in Visual Basic. You can set the property, but the setting has no effect.

To send color output to a color printer, you must use the Windows API function call SetTextColor() instead of the ForeColor property of the Printer object.

Do the following to print "Hello" in all of the 16 QBColors:

1. Start 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. In the Form1 global module, add the following:

```
' Enter the following Declare statement on one, single line:
Declare Function SetTextColor Lib "GDI" (ByVal hDC As Integer,
   ByVal crColor As Long) As Long
```

3. In the Form1 Form_Click event procedure, add the following code:

```
Sub Form_Click
   For i = 0 to 15
```

```
         x& = SetTextColor(Printer.HDC, QBColor(i))
         Printer.Print "Hello"
      Next i
      Printer.EndDoc
   End Sub
```

4. Press F5 to run the program. Click the form.

The word "hello" will print in 16 different colors.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprint kbprg kbbuglist
KBSubcategory: APrgPrint

BUG: Some Controls Not Printed with PrintForm in Windows 3.1

Article ID: Q84471

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

In Windows version 3.1, if you print a Visual Basic form to a printer by either selecting Print from the File menu during design time or by using the PrintForm method during run time, some of the controls on the form may not be not be printed (such as the frame, command button, option button, or check box). This problem is known to occur when using Windows 3.0 video drivers with Windows version 3.1. The problem is also known to occur with third-party video drivers that claim to be Windows version 3.1 compatible. The problem does not occur when you run Visual Basic with Windows version 3.0.

WORKAROUND
==========

To overcome this problem, delete the old video driver and install the new Windows version 3.1 compatible driver. This can be done through Windows Setup (see your Windows version 3.1 manual for details).

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

This problem may result when you install Windows version 3.1, because some of the Windows version 3.0 video drivers may not be updated.

Steps to Reproduce Problem

To reproduce the problem, do the following (using a Windows version 3.0 video driver with Windows version 3.1):

1. Start 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 controls to Form1 such as frame, command button, check box, and option button.

3. From the File Menu, choose Print.

4. In the Print dialog box, select Current and Form.

5. Choose the OK button to start printing.

6. Note that the frame, command button, and option button are not printed.

7. Add the following code to the Form1 Click event:

   ```
   Form1.PrintForm
   ```

8. Press F5 to run the program.

9. Click in the form.

10. Note that the frame, command button, and the check box are not printed.

To overcome this problem, delete the old video driver and install the new Windows 3.1 compatible driver. You can do this through Windows Setup (see your Windows 3.1 manual for details).

Additional reference words: buglist1.00 1.00
KBCategory: kbprint kbprg kbbuglist
KBSubcategory: APrgPrint

BUG: THREED.VBX: Command/Group Push Buttons Show Invalid File

Article ID: Q84553

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

The Picture property for 3-D Command Button and 3-D Group Push Button custom controls can load certain picture formats. However, the 3-D Command Button Load Picture dialog box incorrectly shows *.WMF in the File Name box. This mistakenly indicates that .WMF (Windows metafile) files can be used for pictures. Also, the 3-D Group Push Button Load Picture dialog box for the PictureUp, PictureDn, and PictureDisabled properties incorrectly lists *.WMF and *.ICO in the File Name box. This mistakenly indicates that .WMF and .ICO files can be used for pictures.

RESOLUTION

The 3-D Command Button control Picture property can only use .BMP (bitmap) and .ICO (icon) files. If you attempt to load a .WMF file, the following error message will be displayed:

```
   Only picture formats ".BMP" & ".ICO" supported
```

The 3-D Group Push Button control PictureUp, PictureDn, and PictureDisabled properties can only use .BMP files. If you attempt to load a .ICO or .WMF file, the following error message will be displayed:

```
   Only picture format ".BMP" supported
```

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Generic / Text Only Printer Driver Prints 66 Lines
Article ID: Q87767

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows versions 3.0 and 3.1

SYMPTOMS

Choosing the Generic / Text Only printer drivers in Microsoft Windows versions 3.0 and 3.1 may cause incorrect printing results in Visual Basic for Windows. Visual Basic expects to print 66 lines per page, but the generic printer driver only prints 60 lines per page. This results in six lines being printed on a separate page.

WORKAROUND

To work around the problem, select a different printer driver.

STATUS

Microsoft has confirmed this to be a bug in the Generic / Text Only printer driver included with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available

MORE INFORMATION

When using the Generic / Text Only printer driver, the example below prints 60 lines on the first page, 6 lines on the second page, and then 60 lines on the third page. You may also encounter some lines being overwritten also with the Generic / Text Only driver supplied with Windows version 3.0.

Steps to Reproduce Problem

1. From the Windows Control Panel, choose the Printers icon.

2. From the Printers option, choose the Add Printer button.

3. Select the Generic / Text Only printer driver.

4. Choose the Install button. There is additional help on pages 145-147 of the "Microsoft Windows version 3.0 User's Guide." Note: You may need your Windows disks to install the Generic / Text Only driver.

5. After the Generic / Text Only driver has been installed and is the

   default printer, you can proceed to run a test in Visual Basic.

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

7. Add the following code to the Form_Click event procedure of Form1:

```
   Sub Form_Click ( )
      For i% = 1 to 200
         Printer.Print "This is a test of line number ";i%
      Next i%
      Printer.EndDoc
   End Sub
```

8. From the Run menu, choose Start to run the program.

9. Run the same code, by pressing the F5 function key and then click Form1 once, to run the test. This should produce five pages of text, the first and third pages should have 60 lines of text, while the second and fourth pages will only contain 6 lines of text. The fifth page should be half covered with lines of text. This is where the problem is, Visual Basic sends 66 lines to be printed per page, but the Generic / Text Only printer driver is setup to print only 60 lines. Then the printer driver does a formfeed, after printing the 6 lines on the second page to go on to the third page. The printer driver may also display a problem on some lines of code being overwritten (every fifth line may be overwritten).

Additional reference words: buglist1.00 buglist2.00 1.00 2.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Grid Control Paints Incorrectly When Press PGUP or PGDN

Article ID: Q94296

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows,
  versions 2.0 and 3.0

SYMPTOMS
========

The grid control may paint incorrectly when you press the PGUP or PGDN key. Specifically, when you press the PGDN key to scroll down within a grid control, the data in one column is painted in the next column.

WORKAROUND
==========

This problem does not occur when you use the arrow keys or the mouse to scroll within the grid.

You can work around the problem by refreshing the grid from within a timer. The timer should be activated when the PGUP or PGDN key is pressed. Below are the steps necessary to implement such a workaround:

1. Add a timer control (Timer1) to Form1.

2. Add the following code to the KeyDown event of Grid1:

```
   Sub Grid1_KeyDown (KeyCode As Integer, Shift As Integer)

      'Key codes for the pageup and pagedown keys
      Const VK_PGUP = &H21        'VK_PRIOR
      Const VK_PGDN = &H22        'VK_NEXT

      If KeyCode = VK_PGUP Or KeyCode = VK_PGDN Then
          Timer1.Interval = 1
          Timer1.Enabled = True
      End If

   End Sub
```

3. Add the following code to the Timer1_Timer event:

```
   Sub Timer1_Timer ()
      Grid1.Refresh
      Timer1.Enabled = False
   End Sub
```

When you press the PGUP or PGDN key, the timer event refreshes the grid.

STATUS
======

Microsoft has confirmed this to be a problem in the products listed above.

We are researching this problem and will post more information here in the Microsoft Knowledge Base when it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

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

2. From the File menu, choose Add File (ALT, F, D), and load GRID.VBX into the project if it is not already loaded.

3. Place a grid control (Grid1) on Form1.

4. Set the following properties for Grid1 to these values:

Property	Value
Rows	12
Cols	3
FixedRows	2
FixedCols	1

5. To make the PGUP and PGDN keys applicable, size the grid so that it has fewer than the 12 rows and 3 columns you specified.

6. Add the following code to the Form_Load event of Form1:

```
   Sub Form_Load ()
      Dim i As Integer
      Grid1.Col = 1

      'Fill the first non-fixed column with number from 1 to 11
      For i = 2 To grid1.Rows - 1
         Grid1.Row = i
         Grid1.Text = Format$(i - 1, "0")
      Next
   End Sub
```

7. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program.

8. Set the focus to the grid.

9. Press the PGDN key repetitively until the cursor is at the bottom of the grid. Items from the first non-fixed column (the second column) are incorrectly repeated in the second non-fixed column (the third column).

Additional reference words: buglist2.00 buglist3.00 2.00 3.00 buglist2.00 buglist3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Illegal function call / Division By Zero Errors

Article ID: Q94778

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- The Standard and Professional Editions of Microsoft Visual Basic for MS-DOS, version 1.0
- Microsoft Basic Professional Development System (PDS) for MS-DOS, version 7.1

SYMPTOMS

========

Certain complex numeric expressions may incorrectly cause "Illegal function call" or "Division by zero" errors when run in the interpreter environment of the above mentioned Basic products. This problem only happens on computers that have a math coprocessor.

These errors, however, do not occur with programs compiled using the BC.EXE compiler included with Microsoft Basic Professional Development System for MS-DOS, version 7.1 and the Standard and Professional Editions of Microsoft Visual Basic for MS-DOS, version 1.0

STATUS

======

Microsoft has confirmed this to be a bug with the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

================

To work around this problem, do one of the following:

- Break the complex equation into smaller parts that are evaluated separately.

- Turn off use of the coprocessor with SET NO87=x at the DOS prompt (PDS and Visual Basic for MS-DOS only).

- Compile using the alternate math (/FPa) option (PDS and the Professional Version of Visual Basic for MS-DOS only).

The following code reproduces the "Illegal Function Call" error on a computer that has a coprocessor:

```
   test = 1 + (1 + 1 * (1 * (1 + 1 ^ 1)))
```

The following code reproduces the "Division by zero" error on a computer that has coprocessor:

```
   test = 1 + (1 - 1 * (1 + 1 / 1 ^ 1))
```

These are not the only expressions that cause the problem.

Additional reference words: buglist1.00 buglist2.00 buglist7.10 buglist3.00
1.00 2.00 3.00 7.10
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Stack Fault When Move Sets Tiny Width in 2-Item Combo Box

Article ID: Q95197

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0

SYMPTOMS
========

An Application Error saying that Visual Basic caused a stack fault occurs when you click the down arrow of a combo box if the combo box contains two items and you set the Width property of the combo box to less than 378 from within a Move method. The number it takes to cause the problem depends on your current video mode. This example uses a 1224 by 768 driver. The lower your resolution, the higher the number must be to prevent the Application Error.

WORKAROUND
==========

To work around this problem, set the width of the combo box to 377 in design mode, and don't set it from within a Move method. As another alternative, you can remove one of the two items in the Combo Box.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic versions 2.0 and 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

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

2. Add a combo box (Combo1) to Form1.

3. Add the following code to the Form1_Load event:

```
Sub Form_Load()
  Combo1.additem "Item 1"
  Combo1.additem "Item 2"
  Combo1.Move 100, 100, 377 ' Postion 100, 100, with a width of 382
End Sub
```

4. From the file menu, choose Run to run the program.

5. Click the down arrow of the combo box.

This results in an Application Error stating a stack fault occurred.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: GPF/UAE If Multi-Select Controls w/ No Common Properties

Article ID: Q95430

The information in this article applies to:

 - The Standard and Professional Editions of Microsoft Visual Basic
   for Windows, versions 2.0 and 3.0

SYMPTOMS
========

A general protection (GP) fault or unrecoverable application error (UAE) may occur when you select multiple custom controls that have no properties in common.

For example, if you add the VBSQL.VBX custom control from the Microsoft SQL Server Programmer's Toolkit for Visual Basic and then select it and the Timer control while holding down the CTRL key, you will encounter a GP fault or UAE.

CAUSE
=====

The problem occurs because there are no properties in common between the Timer control that comes with Visual Basic and the VBSQL.VBX control. This usually isn't a problem because most custom controls contain at least the Tag property. There are only a few exceptions.

STATUS
======

Microsoft has confirmed this to be a bug in both the Standard and Professional Editions of Microsoft Visual Basic versions 2.0 and 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00 GPF multiselect
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Type Mismatch Error If Use VAL Function on Big Hex Value
Article ID: Q95431

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Standard and Professional Editions of Microsoft Visual Basic for MS-DOS, version 1.0
- Microsoft Basic Professional Development System (PDS) for MS-DOS, version 7.1
- Microsoft QuickBASIC for MS-DOS, version 4.5

SYMPTOMS
========

Using the VAL function on a large hexadecimal number (greater than or equal to the hexadecimal value 80000000) embedded in a string can incorrectly cause a "Type mismatch" error. This occurs only when the hexadecimal number contains an ampersand (&) at the end of the string.

WORKAROUND
==========

To reproduce the problem run the following code:

```
   PRINT VAL("&H80000000&")
```

You get a "Type mismatch" error. To prevent the error, remove the last ampersand (&) character.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist2.00 buglist1.00 buglist4.50 buglist7.10 buglist3.00 1.00 2.00 3.00 4.50 7.10
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOther

BUG: Stack Fault May Occur If Trapping Divide By Zero
Article ID: Q95499

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows,
  versions 2.0 and 3.0

SYMPTOMS
========

When trapping a divide by zero or divide overflow error (error numbers 11 and 6 respectively) in a Visual Basic program, you may receive a stack fault if an MS-DOS session is also running. In this situation, the computer may also hang (stop responding) or automatically reboot.

CAUSE
=====

This problem is caused by the Windows mathematics exception handling, not by
Microsoft Visual Basic.

WORKAROUND
==========

The only way to avoid this problem is to terminate all MS-DOS sessions before running a Visual Basic application that traps divide by zero or divide overflow errors.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Windows version 3.1. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start an MS-DOS session in Windows. If the MS-DOS session appears full screen, press ALT+ENTER to make it a windowed session.

2. Minimize the MS-DOS window.

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

4. Add the following code to the Form_Click event procedure of Form1:

```
   Sub Form_Click ()
      On Error Resume Next
      Top:
         x% = DoEvents()
         y% = 1 \ 0  'This will cause a division by zero error
      GoTo top
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program.

6. Click in the Form1 form. You may receive a stack fault here. if not, continue with step 7.

7. Double-click the minimized MS-DOS session icon to restore it.

You should receive the message "VB caused a Stack Fault in module VB.EXE at 0001:0009."

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtRun

BUG: GPF When Close Form That Contains a Single MCI Control

Article ID: Q95500

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
---------------------------------------------------------------------
```

SYMPTOMS
========

When you have a single MCI control on a form and you set the hWndDisplay property to the form's hWnd, you will receive a general protection (GP) fault upon closing Form1 through the System Control of Form1. This problem does not occur when you have a second control on Form1 in which you set the HwndDisplay property to the hWnd of the other control.

WORKAROUND
==========

Here's an example that shows how to work around the problem. The code listed below places a picture box on Form1, changes the BoarderStyle to '0' (None), and then places an MCI control on Form1:

```
   Sub Form_Load()
      MMControl1.FileName = "c:\vb\samples\mci\mcitest.mmm"
        '** file in the ..\samples\mci directory of VB 2.0
      MMControl1.hWndDisplay = Picture1.hWnd
        '** note the picture's hWnd is used in place of the form's.
      MMControl1.Command = "Open"
   End Sub

   Sub Form_Unload()
      MMControl1.Command = "Close"
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start VB.EXE.

2. Choose Add File... from the File menu and add the MCI.VBX file.

3. Place an MCI control on Form1.

4. Place the following code in the Form_Load event procedure of Form1:

```
   Sub Form_Load()
      MMControl1.FileName = "c:\vb\samples\mci\mcitest.mmm"
         '** file in the ..\samples\mci directory of VB 2.0
      MMControl1.hWndDisplay = Form1.hWnd
         '** docerr DisplayHwnd on page 248 of Professional Features
      MMControl1.Command = "Open"
   End Sub
```

5. Place the following code in Form_Unload event of Form1:

```
   Sub Form_Unload()
      MMControl1.Command = "Close"
   End Sub
```

6. Press the F5 key to run the example, which may result in a GP fault when you try to close Form1's System Control box. The GP fault address is 0001:2817.

Note this example and any example of using the MCI control can be run only in Windows version 3.1 or in Windows version 3.0 with Multimedia Extensions. You need add the following line to the Multimedia Extensions section ([mci extensions]) of your WIN.INI file:

```
   MMM=MMMovie
```

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Neg ScaleHeight Resizes Control When Form Saved as ASCII

Article ID: Q95513

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0

SYMPTOMS
========

If you set the ScaleHeight or ScaleWidth property of a container to a negative value, the Height or Width property of all child controls are saved incorrectly if the form is saved in ASCII format. When you re-load a form or its project that was previously saved in ASCII format, it may look like controls on the container have been removed. Actually, the child controls still exist, but their Height and Width properties were saved incorrectly, which results in significantly smaller controls.

WORKAROUND
==========

To work around the problem:

1. Resize the controls to their original size by using the mouse. You must use the mouse; you cannot resize the controls by changing the Height and Width properties in the Property window. Click the lower right-hand corner of the control and drag it down or to the right to make the control taller or wider, respectively.

2. Save the form in binary format. From the File menu, choose Save Project (ALT, F, V) and clear the Save as Text check box option.

STATUS
======

Microsoft has confirmed this to be a bug in both the Standard and Professional Editions of Microsoft Visual Basic versions 2.0 and 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

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. From the Properties window, set the following properties of Form1:

```
   ScaleMode:      0   (User)
   ScaleHeight: -100
   ScaleWidth:  -150
   ScaleTop:    -100   Sets upper left hand corner coordinates
   ScaleLeft:   -150   of Form1 to (-150,-100)
```

3. Add a command button (Command1) to Form1.

4. From the Properties window, set the properties of the Command1 button as follows to place the command button in the middle of the form.

   ```
   Top:  -150
   Left: -200
   ```

5. From the File menu, choose Save Project (ALT, F, V). Select the Save as Text option and save the form using the default name of Form1. Save the project (Project1) using the default name.

6. From the File menu, choose Open Project (ALT, F, O). In the Files box, select PROJECT1.MAK.

7. From the Window menu, choose Project (ALT, W, R). Using the mouse, click View Form in the Project window. Form1 displays, and you can see that the Command1 button is significantly smaller, making it difficult to pinpoint where it is.

8. Using the mouse, click Form1 to change the focus to Form1.

9. Press the Tab key to move the focus to the command button. Now Command1 becomes visible and the Properties window shows its properties.

You can resize or move the command button by using the mouse. However, if you attempt to set the Height property of Command1 to a positive value, Visual Basic incorrectly changes the property to its minimum value. The minimum value for the Height property is based on the FontName and FontSize properties.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOptTips

BUG: Stack Fault When Move Makes Combo Box Width Too Small
Article ID: Q95830

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

A Stack Fault results if a Move method changes the Width property of a
combo box containing two to eight items to a small value.

The optional third parameter to the Move method changes the width property
of the control to which the method applies. Applying the method to the
combo
box with a third parameter of less then 240 when the scale mode is set to
twips, produces a Stack Fault Application Error halting the execution of
your application.

WORKAROUND
==========

Changing the Width property, by using the Move method or by setting the
property directly, to a value as small as 240 practically eliminates the
functionality of the control. At this width, the combo box is barely wide
enough to view the drop-down button. Hence no entries in the combo box are
visible to the user.

If want your application to move the control to a position where the user
can not view the control at that instant, use one of these techniques:

1. Set the Visible property of the combo box to False.
2. Set the Top and Left properties of the combo box to position the control
   outside the visible region of the Form.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

The Application Error dialog box indicates that Visual Basic caused the
Stack Fault in USER.EXE. However the address differs depending on the
version of Visual Basic. In version 2.00, the address is 0007:0CA3. In
version 1.00, the address is 0001:707A.

Steps to Reproduce Problem

----

1. Start 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 a combo box (Combo1) to Form1.

3. Place the following code in the Form_Click event procedure:

```
   Sub Form_Click ()
      combo1.AddItem "Item 1"
      combo1.AddItem "Item 2"  ' Add two items to combo1
      combo1.Move 0, 0, 240    ' New position = (0,0);Width = 240
   End Sub
```

4. From the Run menu, choose Start (ALT, R, S) to run the program.

5. Using the mouse, click Form1. At this point, the combo box moves to the upper-left corner of Form1 and its width changes to 240 twips (The default ScaleMode).

6. Using the mouse, click Form1 again. An Application Error dialog appears stating the following:

```
      VB caused a Stack Fault in module USER.EXE at 0007:0CA3
```

   Running Visual Basic version 1.00 displays a similar message with an address of 0001:707A.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtRun

BUG: Unable to Edit LinkNotify Event If Control Has Long Name

Article ID: Q97027

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

Visual Basic version 2.0 does not allow you to edit the LinkNotify event procedure of a Label, Picture Box, or Text Box control if the control has a 30-character Name property.

CAUSE

The LinkNotify event, an event new in Visual Basic version 2.0, became the longest (10 characters) event procedure name for Label, Picture Box, and Text Box controls. In version 1.0, the longest event procedure for these controls was nine characters long.

The maximum length of the Name property (CtlName property in Visual Basic version 1.0) is directly related to the length of the control's longest event procedure, so the maximum length of the Name property for Label, Picture Box, and Text Box controls in Visual Basic version 2.0 is one character less than it is in Visual Basic version 1.0.

Therefore, if you load a Visual Basic version 1.0 project into Visual Basic version 2.0 and a Label, Picture Box, or Text Box control has a 30-character CtlName property, you won't be able to edit the LinkNotify event in the Visual Basic environment until you reduce the length of the Name property.

WORKAROUND

Reduce the length of the Name property by one or more characters.

STATUS

Microsoft has confirmed this to be a bug in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

The maximum length of event procedures names is limited to 40 characters including the control name, the underscore, and the event name. The Name property therefore has a maximum length that varies depending on the events supported by the control.

Steps to Reproduce Problem
----

1. Start Visual Basic version 1.0, or from the File menu, choose New
   Project (ALT, F, N) if Visual Basic is already running. Form1 is created
   by default.

2. Add a Text box (Text1) to Form1.

3. Set the CtlName property of Text1 to the following 30-character name:

   Text56789012345678901234567890

4. From the File menu, choose Save Project (ALT, F, V). Save the form and
   project with their default names, Form1 and Project1.

5. From the File menu, choose Exit to close Visual Basic version 1.0.

6. Start Visual Basic version 2.0.

7. From the File menu, choose Open Project (ALT, F, O) and select Project1.
   Two dialog boxes will appear stating that Form1 and Project1 are saved
   in an older format and will be saved in new format when you save the
   project. Choose the OK button on both dialog boxes.

8. From the View menu, choose Code (ALT, V, C) to open a code window for
   Form1.

9. From the Object List, select Text56789012345678901234567890.

10. From the Procedures List, try to select LinkNotify.

At this point, the Visual Basic environment will not allow you to select
LinkNotify. It returns you to the previously displayed event procedure.

Additional reference words: buglist2.00 1.00 2.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: ODBC Getchunk Method on Non-Memo Field Causes GPF/UAE

Article ID: Q97082

The information in this article applies to:

 - The Professional Edition of Microsoft Visual Basic for Windows,
   version 2.0

SYMPTOMS
========

Attempting to use the GetChunk method on a Text field or any field that has a data type other than Memo results in an unrecoverable application error (UAE) or a general protection (GP) fault.

CAUSE
=====

The GetChunk method returns a string that represents all or a portion of a Memo field and only a Memo field in a specified dynaset.

WORKAROUND
==========

To avoid the problem, use code to ensure that the field is a Memo field before you call the GetChunk method. For example, replace the following line shown in step 2 of the More Information section of this article:

```
   string1$ = ds(ds.Fields(NonMemoFieldNum%).Name).GetChunk(0, 50)
```

with this code:

```
   If ds.Fields(NonMemoFieldNum%).type = 12 Then
      string1$ = ds(ds.Fields(NonMemoFieldNum%).Name).GetChunk(0, 50)
   End If
```

STATUS
======

Microsoft has confirmed this to be a bug in the Professional Edition of Microsoft Visual Basic version 2.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start the Professional Edition of VB.EXE with ODBC support already installed.

2. Add the following code to the Form_Click event procedure of Form1:

```
   Form_Click ()
      Dim db As database
      Dim ds As dynaset

      ServerName$ = "aServerName"  ' Provide the name of a real server.
      DBName$ = "aDatabase"        ' Name of a database on the server.
      TableName$ = "aTable"        ' Name of a table in the database.
      UserName$ = "aUser"          ' login id
      PW$ = ""                     ' password
      NonMemoFieldNum% = 1  ' This could be any field in the table that
                            '  is not of type "Memo".

      'Connect to the SQL database
      Connect$ = "UID=" + UserName$ + ";PWD=" + PW$ + ";DBQ=" + DBName$

      Set db = OpenDatabase(ServerName$, False, False, Connect$)

      Set ds = db.CreateDynaset(TableName$)
      ' GP fault occurs on the following line:
      string1$ = ds(ds.Fields(NonMemoFieldNum%).Name).GetChunk(0, 50)
   End Sub
```

3. Press the F5 key or ALT+R+S, and click Form1.

This results in a GP fault usually at address 0009:08EC in VBODBCA.DLL.

Additional reference words: buglist2.00 2.00 GPF
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: APrgDataODBC

BUG: OLE DataText Prop Doesn't Free Memory When Object Closed
Article ID: Q97136

```
----------------------------------------------------------------------
The information in this article applies to:

- The Professional Edition of Microsoft Visual Basic for Windows,
  version 2.0
----------------------------------------------------------------------
```

SYMPTOMS
========

An OLE destination (OLE client) control (OLECLIEN.VBX) can send data to
the OLE source (OLE server) application by setting the DataText
property, however the memory allocated for this data is not released
until OLECLIEN.VBX is unloaded. The memory is freed when you exit from
the application.

STATUS
======

Microsoft has confirmed this to be a bug in the Professional Edition of
Microsoft Visual Basic version 2.0 for Windows. We are researching this
problem and will post new information here in the Microsoft Knowledge Base
as it becomes available.

MORE INFORMATION
================

Each time an OLE destination object is created and the DataText property
is set, a new private segment is allocated by OLECLIEN.VBX. When working in
the VB.EXE interpreter environment, this segment is deallocated when you
exit from VB.EXE or when you start a new project (ALT+F+N). A Visual
Basic EXE program deallocates this segment when it is unloaded.

The following code uses Microsoft Graph as the OLE source application, but
the memory leak also occurs if OLECLIEN.VBX is used with other OLE source
programs.

To verify that the memory leak occurs, run the code listed below. Then load
a tool like Heap Walker that ships with the Microsoft Windows Software
Development Kit (SDK), and watch the number of private segments allocated
to OLECLIENT change even after the code deletes the OLE objects.

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the
   OLECLIEN.VBX custom control file. The OLE destination (client) tool
   appears in the Toolbox.

3. Place an OLEClient control on Form1.

4. Enter the following code:

```
   Private Sub Form_Click ()
      Const OLE_CREATE_NEW = 0
      Const OLE_UPDATE = 6
      Const OLE_ACTIVATE = 7
      Const OLE_CLOSE = 9
      Const OLE_DELETE = 10

      OleClient1.Class = "MSGraph"
      OleClient1.Protocol = "StdFileEditing"
      OleClient1.ServerType = 1
      OleClient1.Action = OLE_CREATE_NEW
      OleClient1.Action = OLE_ACTIVATE
      OleClient1.Format = "CF_TEXT"   ' MS Graph accepted format

      Title$ = "This is a title" & Chr$(10)

      ' The data for a graph
      Dim Tb As String  ' tab character
      Tb = Chr$(9)
      GraphData$ = "A" & Tb & "3" & Tb & "4" & Tb & "5" & Chr$(10)
      GraphData2$ = "B" & Tb & "9" & Tb & "2" & Tb & "4" & Chr$(10)

      ' Cause a private segment in OLECLIEN to be allocated.
      OleClient1.DataText = Title$ & GraphData$ & GraphData2$

      OleClient1.Action = OLE_UPDATE
      OleClient1.Action = OLE_CLOSE
      OleClient1.Action = OLE_DELETE
   End Sub
```

6. From the Run menu, choose Start.

7. Run a utility such as Heap Walker to list the number of segments
   allocated to OLEClient.

8. Click the form to create and delete an OLE object from Microsoft Graph.

At this point, you'll see that the number of private segments allocated to
OLEClient increases by 1.

Additional reference words: buglist2.00 2.00 MemLeak
KBCategory: kbole kbprg kbbuglist
KBSubcategory: IAPOLE

UPD: GP Fault in KRNL286 When Run EXE on 286 or w/ NT on MIPs

Article ID: Q99251

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0
---------------------------------------------------------------------

SYMPTOMS
========

You may encounter a general protection (GP) fault in KRNL286 at 0001:259F
or 0001:4FEC when you try to run a Visual Basic executable (.EXE) file in
Windows on a 286 computer or in Windows NT on a MIPs computer.

This problem will not occur when running a Visual Basic application from
the Visual Basic design environment on a 286 or MIPs computer.

RESOLUTION
==========

This problem has been fixed in a post-release version of VBRUN300.DLL,
which is available as part of self-extracting file named VBRUN300.EXE from
the Microsoft Software Library.

Download VBRUN300.EXE, a self-extracting file, from the Microsoft Software
Library (MSL) on the following services:

 - CompuServe
      GO MSL
      Search for VBRUN300.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN300.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN300.EXE

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft Visual Basic
programming system for Windows, version 3.0. To correct the problem, obtain
the post-release version of VBRUN300.DLL.

MORE INFORMATION
================

This bug occurs because of a problem with VBRUN300.DLL. The date,
time, size and version number of the VBRUN300.DLL file that leads to
this problem is as follows:
   Date: 04-APR-1993
   Time: 12:00 a.m.
   Size: 394384
   Version: 03.00.0537
```

The date, time, size and version number of the VBRUN300.DLL file that fixes this problem is as follows:

```
   Date: 12-MAY-1993
   Time: 12:00 a.m.
   Size: 398416
   Version: 03.00.0538
```

VBRUN100.DLL & VBRUN200.DLL Also Available in Self-Extracting Files
----

For your convenience, you can also obtain the .DLL files for Visual Basic versions 1.0 (VBRUN100.DLL in VBRUN100.EXE) and 2.0 (VBRUN200.DLL in VBRUN200.EXE). These files are not updates but are provided for your convenience.

Download VBRUN100.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for VBRUN100.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN100.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN100.EXE
```

Download VBRUN200.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for VBRUN200.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN200.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN200.EXE
```

Steps to Reproduce Problem in Visual Basic Version 3.0

----

1. Start 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. From the File menu, choose Make EXE File (ALT, F, k) and use the
   default name of PROJECT1.EXE.

3. Copy PROJECT1.EXE and VBRUN300.DLL to a 286 computer running Windows or
   a MIPs computer running Windows NT.

4. Run PROJECT1.EXE.

A GP fault occurs in KRNL286 at 0001:259F or 0001:4FEC.

Additional reference words: 3.00 GPF softlib update3.00 S14633 S14632
S14631
KBCategory: kbenv kbprg kbbuglist kbfixlist kbfile
KBSubcategory: EnvtRun

BUG: Changing Default Printer Doesn't Effect Printer.Fonts
Article ID: Q99705

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you change the default printer at run time, the Printer.Fonts enumeration is not updated. The Printer.Fonts enumeration is updated only after you print to the new default printer and use the EndDoc method.

WORKAROUND
==========

To work around to this bug, choose one of these techniques:

1. Use Printer.Print "" followed by Printer.EndDoc
2. Call a DLL function which in turn calls the Windows API function EnumFontFamilies or EnumFonts. For a DLL code sample that shows how to enumerate fonts from a DLL, query on the following words in the Microsoft Knowledge Base:

```
      EnumFontFamilies AND EnumFonts
```

A disadvantage in using workaround 1 is that it will always cause a blank page to be ejected. A disadvantage of workaround 2 is that you will need to write a DLL using other Windows programming tools such as Microsoft Visual C++.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

To reproduce this bug, you will need to set up two printer devices for two types of printers. For example, you can set up LPT1 to use an Epson printer driver and LPT2 to use an HP LaserJet printer driver. The default printer will need to be set to one of these devices.

The steps below demonstrate using the Common Dialog custom control to change the default printer. This control is provided with the Microsoft Visual Basic Professional Toolkit version 1.0, the Microsoft Visual Basic Professional Edition version 2.0, and both the professional and standard editions of Microsoft Visual Basic version 3.0.

Steps to Reproduce Problem

1. Start Visual Basic or from the File menu, choose New Project (ALT, F, N) if Visual Basic is already running.

2. Add a common dialog (CMDialog1) control to Form1

3. Add the following code to Form_Click for Form1

```
   Sub Form_Click ()

      Dim i As Integer
      CMDialog1.PrinterDefault = True

      'Show the Printer dialog
      CMDialog1.Action = 5

      Debug.Print Printer.FontCount

   End Sub
```

4. From the Run menu, choose Start (ALT, R, S) or press F5 to run the program.

5. Click Form1. The Printer Dialog is displayed.

6. Choose the OK button to close the dialog. The number of fonts available will be displayed in the Debug Window.

7. Click Form1 again. Select "Setup..." from the Printer dialog. The Printer Setup dialog is displayed.

8. Set the default printer to a different printer and choose the OK button to close the Setup dialog.

9. Choose the OK button on the Printer Dialog to close it.

The same number of fonts found in Step 6 will be displayed in the Debug Window. This demonstrates that Visual Basic did not update the Fonts list. If you step through the fonts in the Printer.Fonts enumeration, you will see the same set of fonts that were available in Step 6.

To see a different number of fonts displayed for the new default printer, from the Run menu, choose End (ALT, R, E) to end the program. Then press F5 to run it again, click Form1, and choose OK on the Printer Dialog.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00 3.00
KBCategory: kbprint kbprg kbbuglist
KBSubcategory: APrgPrint

BUG: GP Fault with Visual Basic DDE Sample & Word for Windows
Article ID: Q99812

The information in this article applies to:

- Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Word for Windows, versions 2.0a, 2.0b, and 2.0c

SYMPTOMS
========

Running the Visual Basic DDE sample with Microsoft Word for Windows may cause a general protection (GP) fault.

STATUS
======

Microsoft has confirmed this to be a problem with Microsoft Word for Windows versions 2.0a, 2.0b, and 2.0c. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Word for Windows (WINWORD.EXE).

2. Start Visual Basic version 3.0 for Windows.

3. From the File menu, choose Open Project (ALT, F, O). Then open the DDE.MAK project from the \VB\SAMPLES\DDE directory.

4. From the Run menu, choose start (ALT, R, S), or press F5. The main form of DDE.MAK is titled DDE Experimenter.

5. From the DDE Experimenter form, select WinWord as the Application and Document1 as the Topic. The Item automatically becomes \Doc.

6. Select the Manual option.

7. Click the Connect button. The caption for the command button will change to Disconnect.

7. Type text into the text box in the Destination Data section of the DDE Experimenter form.

8. Click the Poke button.

9. Select the Automatic option.

At this point, a GP fault occurs in USER.EXE. The address of the GP fault varies depending on the version of Word for Windows. Although the message

indicates that Visual Basic caused the GP fault, the problem is actually
caused by Word for Windows, not Visual Basic.

Additional reference words: 3.00 WinWord 2.00
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: IAPDDE

BUG: Wrong Menu Click Event After Hiding Menu
Article ID: Q99872

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The wrong menu Click event is executed after hiding and showing menu items in Visual Basic.

CAUSE
=====

This problem occurs when a menu is made invisible before another menu item is made visible.

WORKAROUND
==========

Change the order followed to make menus visible and invisible. For example replace the following code (listed in step 4 in the More Information section below):

```
   Sub Command1_Click ()
      MnuFile.Visible = 0
      MnuEdit.Visible = -1
   End Sub

   Sub Command2_Click ()
      MnuEdit.Visible = 0
      MnuFile.Visible = -1
   End Sub
```

with this code:

```
   Sub Command1_Click ()
      MnuEdit.Visible = -1
      MnuFile.Visible = 0
   End Sub

   Sub Command2_Click ()
      MnuFile.Visible = -1
      MnuEdit.Visible = 0
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the menu design dialog box of Visual Basic (VB.EXE), create a set of menus using the following table as a guide:

Caption	CtlName (or Name)	Level	Visible
&File	MnuFile	1	False
&New	MnuFileNew	2	True
&Edit	MnuEdit	1	False
&Copy	MnuEditCopy	2	True

3. Add two command buttons (Command1 and Command2) to the form.

4. Add the following code to your program in the appropriate places:

```
   Sub Command1_Click ()
      MnuFile.Visible = 0
      MnuEdit.Visible = -1
   End Sub

   Sub Command2_Click ()
      MnuEdit.Visible = 0
      MnuFile.Visible = -1
   End Sub

   Sub MnuEdit_Click ()
      Debug.Print "Edit Click"
   End Sub

   Sub MnuEditCopy_Click ()
      Debug.Print "Copy Click"
   End Sub

   Sub MnuFile_Click ()
      Debug.Print "File Click"
   End Sub

   Sub MnuFileNew_Click ()
      Debug.Print "New Click"
   End Sub
```

5. From the Run menu, choose start (ALT, R, S), or press F5.

6. From the Window menu, choose debug (ALT, W, D), or press CTRL+B.

7. Click Command1. You will see the Edit menu on Form1.

8. Click the Edit menu on Form1. Then click the Copy menu. You will
   see Edit Click and Copy Click displayed in the Debug Window.

9. Click Command2. You will now see the File menu in place of the
   Edit menu on Form1.

10. Click the File menu on Form1. Then click the New menu. You will
    see File Click and New Click in the Debug Window.

11. Repeat steps 7 and 8. Instead of seeing Edit Click and Copy Click
    in the Debug Window, you will now see New Click and Copy Click in
    the Debug Window.

The click event for the previously visible menu is being executed
instead of the click event for the currently visible menu.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: MaskedEdit MaxLength Reset to 64 When Mask=""

Article ID: Q99873

```
---------------------------------------------------------------------
The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0
---------------------------------------------------------------------
```

SYMPTOMS
========

When the Mask property of the MaskedEdit custom control is set to two quotation marks (""), the MaxLength property is incorrectly reset to 64. However, the control continues to correctly limit input based on the original MaxLength setting, and you can change the value of MaxLength to establish a different maximum text limit for the control.

WORKAROUND
==========

To work around the problem, store the MaxLength property before setting the Mask property of the MasedEdit custom control. Then reset the MaxLength setting after setting the Mask property.

For example, replace the code shown in the Command2_Click event procedure in step 3 of the More Information section below with this code:

```
   Sub Command2_Click ()
      Dim ml As Integer
      'Store the current MaxLength property value
      ml = maskededit1.MaxLength
      maskededit1.Mask = ""
      maskededit1.Text = ""
      'Restore the MaxLength property value since
      'it has incorrectly been reset to 64
      maskededit1.MaxLength = ml
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

If you set the mask property to "" at run time the MaxLength property is incorrectly set to 64, but the amount of text you can enter is still limited by the original MaxLength setting.

Steps to Reproduce Problem
----

1. Start 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 three Command buttons (Command1, Command2, and Command3) to Form1.

3. Add the following code to your program in the appropriate places:

```
Sub Command1_Click ()
   Debug.Print "MaskedEdit1.maxlength", maskededit1.MaxLength
   Debug.Print "Text length", Len(maskededit1.Text)
End Sub

Sub Command2_Click ()
   maskededit1.Mask = ""
   maskededit1.Text = ""
End Sub

Sub Command3_Click ()
   Debug.Print "MaxLength set to 10"
   maskededit1.MaxLength = 10
End Sub
```

4. From the Run menu, choose start (ALT, R, S), or press F5.

5. From the Window menu, choose debug (ALT, W, D) or press CTRL+B. The Debug Window will be displayed.

6. Click Command1. You will see the current Maxlength value of 64 and the current text length of 0 displayed in the Debug Window.

7. Click Command3 to set MaxLength to 10. This is verified in the Debug Window. Type text into the MaskedEdit1 control. Notice that you are allowed to enter a maximum of 10 characters.

8. Click Command1. The Debug Window shows that Maxlength is set to 10. The current text length will reflect the number of characters you typed into the MaskedEdit1 Control.

9. Click Command2. This sets the mask property to "", and clears the text in the MaskedEdit1 control.

10. Click Command1 to see that the Maxlength property is now incorrectly set to 64. Type text into the MaskedEdit1 control, and note that you allowed to enter a maximum of 10 characters.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

UPD: GENERIC Sample Not Provided with Visual Basic
Article ID: Q99888

The information in this article applies to:

- Professional Edition of Visual Basic for Windows, version 3.0

SYMPTOMS
========

Appendix E of the Control Development Guide in the "Microsoft Visual Basic Version 3.0 Professional Features Book 1" manual refers to a sample called GENERIC that it says is in the \SAMPLES\GENERIC subdirectory of Visual Basic. However, this sample was not provided with Visual Basic.

RESOLUTION
==========

You can get the GENERIC sample files by downloading a self-extracting file (GENERIC.EXE) from the Microsoft Software Library. After downloading the file, run it to obtain the GENERIC sample files.

Download GENERIC.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for GENERIC.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download GENERIC.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get GENERIC.EXE
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft Visual Basic programming system version 3.0 for Windows. This problem can be corrected by downloading the GENERIC sample files.

Additional reference words: 3.00 update3.00 softlib S14634
KBCategory: kbprg kbbuglist kbfixlist kbfile
KBSubcategory: PrgOther

BUG: Overflow Error When CurrentX Or CurrentY Greater Than 32K
Article ID: Q100190

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows,
  versions 2.0 and 3.0

SYMPTOMS
========

An Overflow error results if you attempt to set CurrentX or CurrentY to a value greater than 32,767 while the current ScaleMode is set to Twips. When using another ScaleMode such as pixels, the same problem occurs if the conversion of the CurrentX or CurrentY value to twips is greater than 32,767.

However, when you use the Print method (or other graphics method) you can correctly cause the value of CurrentX or CurrentY to exceed 32,767 when the ScaleMode is set to twips.

CAUSE
=====

When CurrentX or CurrentY is set explicitly, Visual Basic incorrectly converts the value using the current scale mode to twips. If the result of the conversion to twips is greater the 32,767, an Overflow error occurs. For example, if the ScaleMode is set to Pixels, CurrentX and CurrentY cannot exceed approximately 2731 pixels if the twips per pixel ratio is 12 because 12 times 2731 is 32,772 which is greater than 32767.

When setting CurrentX or CurrentY, Visual Basic should convert the value using the current ScaleMode to pixels rather than twips before comparing the result to 32,767. As a result of this bug, CurrentX and CurrentY are each restricted to a limit 12-14 times smaller (depending on TwipsPerPixelX or TwipsPerPixelY) than they should be.

WORKAROUND
==========

To work around the problem, call the Windows API functions:

- Call TextOut to control the position of text in a picture box or a form.
- Call MoveTo and LineTo to control the position of a line.
- Call other appropriate Windows API functions to position the output for other graphics methods such as the circle method.

An example is shown in the More Information section below.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Because the ratio of twips per pixel varies from one device (or screen resolution) to another, you will need to calculate the limit for the device you are using. To calculate the exact pixel limit of CurrentX, divide 32768 by Screen.TwipsPerPixelX. To calculate the limit of CurrentY, divide 32768 by Screen.TwipsPerPixelY. To find the limit of CurrentX and CurrentY for your printer, use the Printer object in place of the Screen object in the calculations above.

Example for Using API Calls as Workaround

The following example shows how to use the three API calls TextOut, MoveTo, and LineTo to work around the problem. Note that when you call Windows API functions to print or draw, all X and Y coordinates are measured in pixels regardless of the current ScaleMode setting.

1. Start 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 declarations to the General section of Form1

```
' Enter the following Declare statement on one, single line:
Declare Function TextOut Lib "GDI" (ByVal hDC As Integer,
   ByVal X As Integer, ByVal Y As Integer, ByVal lpString As String,
   ByVal nCount As Integer) As Integer

' Enter the following Declare statement on one, single line:
Declare Function MoveTo Lib "GDI" (ByVal hDC As Integer,
   ByVal X As Integer, ByVal Y As Integer) As Long

' Enter the following Declare statement on one, single line:
Declare Function LineTo Lib "GDI" (ByVal hDC As Integer,
   ByVal X As Integer, ByVal Y As Integer) As Integer
```

3. Add the following code to the Form_Click event

```
Sub Form_Click ()
   X1% = 100
   Y1% = 100
   X2 %= 200
   Y2 %= 200

   retvaL& = TextOut(FORM1.hDC, 100, 100, "ONE LINE", 8)

   retvaL& = MoveTo(FORM1.hDC, X1%, Y1%)

   retvaL& = LineTo(FORM1.hDC, X2%, Y2%)

End Sub
```

4. From the Run menu, choose start (ALT, R, S), or press F5 to run the program.

5. Click the form, and you will see the words "ONE LINE" on the form and a diagonal line from the upper left to the lower right. The line starts at the X1 and Y1 coordinates given in the MoveTo API call and ends at the X2 and Y2 coordinates given in the LineTo API call. The words "ONE LINE" should appear 100 pixels from the top and 100 pixels from the left. Note that TextOut may be used without MoveTo because TextOut gives its own coordinates. However using LineTo without using MoveTo results in a line stating from the current output position.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: VB Pro Setup Fails to Correctly Associate .HLP Files

Article ID: Q100191

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic programming system
  for Windows, version 3.0

SYMPTOMS

If you click a file with the .HLP extension from File Manager, you may receive this message:

```
   Cannot Run Program. There is no application associated with this
   file. Choose Associate form the File menu to create an association.
```

CAUSE

The Setup program in the Professional Edition of Visual Basic version 3.0 for Windows adds the following problem line to the extensions section of the WIN.INI file if no association for .HLP file currently exists:

```
   HLP=D:\WINDOWS\SETUPWIZ.INI ^.HLP
```

If there is already an entry for the HLP file extension in the WIN.INI file no change is made by the setup program.

WORKAROUND

Locate the following line in the WIN.INI file in the \WINDOWS directory:

```
   HLP=D:\WINDOWS\SETUPWIZ.INI ^.HLP
```

Replace it with this line:

```
   HLP=WINHELP.EXE ^.HLP
```

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist3.00 3.00 Help
KBCategory: kbsetup kbbuglist
KBSubcategory: Setins

BUG: Out of Memory Error on Show Next from Debug Menu
Article ID: Q100192

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

If you choose Show Next Statement from the Debug menu when you are not stepping through code, Visual Basic gives you an "Out of Memory" error message.

CAUSE
=====

Visual Basic incorrectly enables the Show Next Statement choice in the Debug menu when you are not in single-step mode. This menu choice should be enabled only when you are stepping through code.

WORKAROUND
==========

Avoid using the Show Next Statement option when you are not single stepping through code. This option should not be available when you are not single stepping through code.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the Run menu, choose Start (ALT, R, S), or press F5.

3. From the Run menu, choose Break (CTRL BREAK).

4. From the Debug menu, choose Show Next Statement (ALT D W).

Visual Basic will display an "Out of Memory" error message.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist

KBSubcategory: PrgCtrlsStd

BUG: 3D Button Loses 256-Color Palette When Load 2nd Bitmap
Article ID: Q100193

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

If a second 256-color bitmap is loaded in another control after loading a 256-color bitmap in a 3D command button, the palette on the 3D command button is not restored.

CAUSE
=====

The 3D command button control that is part of THREED.VBX does not restore its own palette. Instead, it uses the current system palette when a new 256-color bitmap is load into another control in the project. In effect this causes the 3D command button to use the palette of the new bitmap.

WORKAROUND
==========

To work around this problem, force the current system palette to be the palette used by the 3D command button and refresh the 3D command button. For example, make the following changes to the Picture2_Click event procedure listed in step 4 of the More Information section:

```
   Sub Picture2_Click ()
      Picture2.Picture = LoadPicture("c:\vb3\rainbow.dib")

     ' Add the following two lines to force the picture that has
     ' the same palette as Command3d1 to the top of the ZOrder:
     Picture1.ZOrder 0
     Command3d1.Refresh

   End Sub
```

Using the ZOrder method with zero as an argument moves Picture1 to the top of the ZOrder. This makes the palette for Picture1 the current system palette. Because Picture1 and Command3d1 have the same bitmap loaded, you can clear up the problem by forcing the palette of Picture1 to be the system palette and refreshing the Command3d1 control.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Steps to Reproduce Problem

----

1. Start 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. From the File menu, choose Add File (ALT F, A) and add THREED.VBX to your project.

3. Add two picture boxes (Picture1 and Picture2) and one 3D command button (Command3d1) to the project.

4. Add the following code to your program in the appropriate places:

```
   Sub Command3D1_Click ()
      Command3d1.Picture = LoadPicture("c:\windows\256color.bmp")
   End Sub

   Sub Picture1_Click ()
      Picture1.Picture = LoadPicture("c:\windows\256color.bmp")
   End Sub

   Sub Picture2_Click ()
      Picture2.Picture = LoadPicture("c:\vb3\rainbow.dib")
   End Sub
```

5. From the Run menu, choose start (ALT, R, S), or press F5.

6. Click Picture1.

7. Click Command3d1. Picture1 and Command3d1 should now contain the same bitmap image.

8. Click Picture2. Notice that the bitmap in Picture1 has maintained its palette and the bitmap in Command3d1 has lost its original colors.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus APrgGrap

BUG: Grid Control Repaints When Another Form Is Made Active
Article ID: Q100195

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

If you activate another form while a form containing a Grid control is showing, the Grid repaints itself.

CAUSE
=====

When the grid loses focus, it automatically repaints the entire grid. The grid should only paint the section of the grid that was covered or changed -- not the entire grid -- when it loses focus.

WORKAROUND
==========

There is no known workaround at this time.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File (ALT, F, A) and add GRID.VBX to the project and add a grid (Grid1) control to Form1.

3. From the File menu, choose New Form (ALT F, F). Form2 is created.

4. Add the following code to the Form_Load event procedure of Form1:

```
   Sub Form_Load ()

      Grid1.Rows = 20
      Grid1.Cols = 8

      'Initialize the grid with random data
      For I = 0 To 19
         Grid1.Row = I
         For J = 0 To 7
            Grid1.Col = J
            Grid1.Text = Format$(I) + Format$(J)
         Next J
      Next I

      Form2.Show

   End Sub
```

5. From the Run menu, choose start (ALT, R, S) or press F5.

6. Position Form2 to cover a portion of the grid, click back and forth between the two forms, and notice that the grid is repainted each time Form2 is activated.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Unload in 3D GroupPush Button Causes GP Fault

Article ID: Q100327

The information in this article applies to:

- Professional Edition of the Microsoft Visual Basic Programming System for Windows, versions 2.0 and 3.0
- Professional Toolkit for Microsoft Visual Basic Programming System for Windows, version 1.0

SYMPTOMS

A general protection (GP) fault occurs when you place an Unload statement in the GroupPush3D1_Click event procedure of the THREED.VBX custom control. A GP fault also results, but at a different address, when you use the THREED.VBX custom control shipped with the Professional Edition of Visual Basic version 3.0 for Windows in a Visual Basic version 2.0 or 1.0 application.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the File menu, choose Add File. In the Files box, select the THREED.VBX custom control file. The six 3D controls appear in the Toolbox. Add Form2 to the project by choosing New Form from the File menu.

3. Select the GroupPush3D button tool (with the letters R and B on it) from the Toolbox, and draw it on Form1.

4. Next double-click or press F7 to get to the GroupPush3D1_Click event procedure. Place the following code in this event procedure:

```
Sub GroupPush3D1_Click (Value As Integer)
   Unload Form1       '** result in 3.0, GPF 001D:09C0
   Form2.Show         '** result in 2.0, GPF 003B:09AB
                      '** result in 1.0, GPF 0057:0040

   '** Or
```

```
      '** Form2.Show    '** result in 3.0, GPF 001D:09BD
      '** Unload Form1  '** result in 2.0, GPF 003B:09A8
                        '** result in 1.0, GPF 005C:0629
   End Sub
```

5. To run the example, click the Play button, press the F5 key, or choose Start from the Run menu. Then click the GroupPush3D button. If you get an error, choose the close button, this will result in a GP fault at a specific address.

Additional reference words: buglist3.00 1.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Referencing Data Object Gives Error: Object not an Array

Article ID: Q100367

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

An "Object not an Array" error results when you reference a field of a dynaset, table, or snapshot object in a form or module that does not contain a Set statement for that dynaset, table, or snapshot. The error occurs when Visual Basic attempts to compile your program.

CAUSE
=====

This error is caused by a parsing problem in the Visual Basic programming environment. The Visual Basic parser does not recognize the object type because there is no Set statement in the same form or module.

WORKAROUND
==========

Add a dummy Sub procedure to each form or module, and use a Set statement that refers to the global database / table / dynaset in a meaningful way (for example, Set myds = db.CreateDynaset(...) not set myDs = myDs). Give the Sub procedure a name like 'AAAAA_Fix_Parser' so it will be the first code parsed in that form or module. Make sure the dynaset set in the dummy Sub procedure is the exact same dynaset that is causing the problem.

When adding more than one dummy Sub procedure to a project, give each Sub procedure a different name (AAAAA1, AAAAA2, and so on) to avoid name collisions that could complicate your existing problem.

For example, use the following dummy procedure if MyDs is the dynaset causing the problem:

```
   Sub AAAAAA_Fix_Parser
      Set MyDs = MyDB.CreateDynaset("...")
   End Sub
```

You never need to execute the code in the Sub procedure or even call the Sub procedure. Once you add the Sub, the parser will see the Set statement(s) before it tries to parse any other code, so it won't have trouble with the global objects. After adding the Sub procedure, you won't have to tweak the code every time you reload the project; you can do it once and save it.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the File menu, choose New Module (ATL, F, M). Module1 is created.

3. Add a text box (Text1) to Form1.

4. Add the following code to the General section of Module1

```
   Global MyDs As Dynaset
```

5. Add the following code to Module1

```
   Sub main ()
      Dim MyDB As Database
      Dim SQLStmt As String
      Const DB_READONLY = 4   ' Set constant.
      Set MyDB = OpenDatabase("BIBLIO.MDB")   ' Open database.

      ' Set text for the SQL statement.
      SQLStmt = "SELECT * FROM Publishers WHERE State = 'NY'"

      ' Create the new Dynaset.
      Set MyDs = MyDB.CreateDynaset(SQLStmt, DB_READONLY)

      form1.Show
    End Sub
```

6. Add the following code to the Form_Load event procedure of Form1:

```
   Sub Form_Load ()
      Text1.Text = MyDs("state")
   End Sub
```

7. From the Options menu, choose Project (ALT, O, P). The Projects Options dialog is displayed.

8. From the Project Options dialog, set the Start Up Form to Sub Main and choose OK.

9. From the Run menu, choose start (ALT, R, S) or press F5.

You will get the error "Object not an Array" on the following line:

```
   Text1.text = MyDs("state").
```

Additional reference words: buglist3.00 3.00

KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: APrgDataIISAM PrgCtrlsStd

UPD: New XBASE Driver Available That Fixes Several Problems

Article ID: Q100514

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SUMMARY
=======

A new XBase IISAM driver XBS110.DLL version 1.10.0002 is available. This driver fixes several bugs documented below. It is the same driver that is provided with Microsoft Access version 1.10.

To obtain the new driver, download XBS110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for XBS110.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download XBS110.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get XBS110.EXE
```

MORE INFORMATION
================

If you have Windows for Workgroups, you can use the following steps to get the version number of your XBase driver:

1. Start File Manager

2. Find the XBS110.DLL file, and select it. This file is usually located in the WINDOWS\SYSTEM directory.

3. From the File menu, choose Properties.

The item marked "Version:" is the XBase version number for XBS110.DLL.

Bugs Fixed by XBS110.DLL Version Number 1.10.0002

PROBLEM ID: 2186

    Relates to DBase III

    An update is allowed that violates unique index. Using the XB110.DLL driver that shipped with Visual Basic, it is possible to add multiple records that share the same unique index. The new version of the driver does not allow you to update the database with a record that contains the same unique index value as an existing record.

PROBLEM ID: 2390

Relates to FoxPro 2.5

A general protection (GP) fault occurs when updating the record immediately preceding a record locked by another user. The GP fault occurs in XBS110.DLL at 0002:11DA.

PROBLEM ID: 2418

Relates to DBase III

A unique index is corrupted after an update query. The symptom of this problem is that the first 239 items in the table are not found.

PROBLEM ID: 2432

Relates to DBase III, IV and Fox Pro 2.0, 2.5

SeekEQ on NULL returns first non-null record when there are no NULL records in the column.

PROBLEM ID: 2457

Relates to: FoxPro 2.5

Attempting to update a record results in a GP Fault in XBS110.DLL at 0013:144A when the IDX index type is used.

PROBLEM ID: 2487

Relates to FoxPro 2.5

A GP fault in XBS110.DLL occurs at 001A:05F6 when using INSERT INTO on the same table as the FROM clause uses -- that is, when copying records from a table into itself.

PROBLEM ID: 2511

Relates to FoxPro 2.0 and 2.5

A GP fault in XBS110.DLL occurs at 0002:11DA when inserting the 98th record in table that has one index.

Additional reference words: 3.00 update3.00 softlib S14644 GPF
KBCategory: kbenv kbprg kbbuglist kbfixlist kbfile
KBSubcategory: EnvtRun

BUG: GPF in Some Video Drivers When Load RLE Bitmaps > 20K

Article ID: Q100610

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  for Windows, versions 2.0 and 3.0

SYMPTOMS
========

A general protection (GP) fault occurs in some video drivers when an
RLE bitmap file larger than 20K is loaded into a picture box control
or an image control.

CAUSE
=====

This problem is caused by Microsoft Windows, not Visual Basic for
Windows.

WORKAROUND
==========

No workaround is available at this time.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

This problem has been reported with the 8514.DRV driver at address
0007:175D and with the V7VGA.DRV driver at address 0008:1E20. This
problem may also occur with some third-party drivers.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtRun APrgGrap

BUG: Font3D Property Set Incorrectly in THREED.VBX Controls
Article ID: Q100612

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic programming system
  for Windows, version 3.0

SYMPTOMS
========

If a Visual Basic version 2.0 for Windows form file contains THREED.VBX controls with the Font3D property set to a value greater than zero, Visual Basic version 3.0 may incorrectly force other THREED.VBX controls to have the same Font3D property value.

CAUSE
=====

The THREED.VBX custom control for Visual Basic 2.0 does not write a Font3D value to the form file if Font3D = 0. When Visual Basic version 3.0 loads he form, after the Visual Basic environment reads a Font3D value for one control, it gives the same Font3D property value to all the rest of the THREED.VBX controls. In other words, if the last THREED.VBX control loaded is the only one that has a Font3D entry in the form file, none of the other controls are affected.

WORKAROUND
==========

To work around the problem, edit the Visual Basic version 2.0 form files that were saved in ASCII text format to add a Font3D = 0 line to any THREED.VBX controls that do not already have a Font3D entry.

Visual Basic version 2.0 form files that were saved in the binary format can be changed after they are loaded into Visual Basic version 3.0.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The following is an example of a Visual Basic version 2.0 form file that was saved in ASCII format with two 3D command buttons. One button has a Font3D value, and one does not. Note that controls are saved in the form file in the opposite order in which they were created on the form.

```
VERSION 2.00
```

```
Begin Form Form1
   Caption         =   "Form1"
   Height          =   6636
   Left            =   828
   LinkTopic       =   "Form1"
   ScaleHeight     =   6216
   ScaleWidth      =   6420
   Top             =   1152
   Width           =   6516
   Begin SSCommand Command3D2
      Caption         =   "Command3D2"
      Font3D          =   1  'Raised w/light shading
      Height          =   1092
      Left            =   720
      TabIndex        =   1
      Top             =   2640
      Width           =   3012
   End
   Begin SSCommand Command3D1
      Caption         =   "Command3D1"
      Height          =   1212
      Left            =   720
      TabIndex        =   0
      Top             =   840
      Width           =   3012
   End
End
```

Notice that there is not a Font3D setting for Command3D1. If this file were loaded into Visual Basic version 3.0, Command3D1 would have a Font3D value of 1 instead of 0.

To work around the problem, insert the following line between the Caption and Height lines for Command3D1 in the ASCII form file shown above:

```
   Font3D = 0
```

Now, Visual Basic version 3.0 will read the file correctly.

The Visual Basic 3.0 THREED.VBX writes the Font3D property to the form file for every THREED.VBX control regardless of its value.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Data Access Setup Can Give Incorrect Error Message

Article ID: Q100613

The information in this article applies to:

- Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

When adding an SQL server with the Microsoft ODBC Setup program, you may receive the following incorrect message when the server name is actually correct:

```
   The server <your server name> was not found on the network.
   Are you sure you want to use it?
```

CAUSE
=====

The cause of this problem has not yet been determined. We are researching it.

WORKAROUND
==========

Although Visual Basic Data Access Setup generates this incorrect message, Visual Basic still adds the correct information to the ODBC.INI file and the ODBC driver is set up correctly.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Run the Data Access Setup program.

2. Select SQL Server in the Install Drivers dialog box. Then click the OK button. If the ODBC drivers were installed previously, you will get a message box that asks if you want to replace your driver; choose Yes.

3. Select the Add option in the Data Sources dialog box.

4. Select SQL Server in the Add Data Source dialog box, and click the OK button.

4. In the ODBC SQL Server Setup dialog box, type the name of the data source in the Data Source Name field and a valid SQL server name in the Server field.

5. Click the OK button. At this point, Visual Basic may generate a message box with the following text

   The server <your server name> was not found on the network.
   Are you sure you want to use it?

6. Click the Yes button, and the setup will continue as usual.

Additional reference words: buglist3.00 3.00
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: APrgDataIISAM

BUG: Ref to NPV / IRR / MIRR Gives Undefined Functions Error

Article ID: Q101245

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic programming system
  for Windows, version 3.0

SYMPTOMS
========

If you try to run an application that contains a reference to the NPV, IRR, or MIRR financial function, Visual Basic for Windows generates this error:

```
   Reference to undefined Function or Array
```

CAUSE
=====

Visual Basic does not recognized these as Visual Basic functions because they were incorrectly referenced in the financial DLL file (MSAFINX.DLL) that ships with Visual Basic version 3.0.

WORKAROUND
==========

To workaround the problem, declare the NPVC, IRRC, and MIRRC functions located in MSAFINX.DLL and alias them as NPV, IRR, and MIRR respectively. The code provided in the More Information section below demonstrates how to declare and call these functions.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The following example shows how to use the NPV function. It is based on the example shown in the Visual Basic Help menu, but it also includes the declarations for the NPV, IRR, and MIRR financial functions. Without the declarations for these functions, the example will fail, giving a "Reference to undefined Function or Array" error.

Steps to Work Around the Problem
----

1. Start 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 code to the General section of Form1:

```
   ' Enter each Declare statement on one, single line:
   Declare Function MIRRC Lib "MSAFINx.DLL" (values#, ByVal cvalues%,
      ByVal finance#, ByVal reinvest#) As Double
   Declare Function NPVC Lib "MSAFINx.DLL" (ByVal Rate1#, values#,
      ByVal cvalues%) As Double
   Declare Function IRRC Lib "MSAFINx.DLL" (values#, ByVal cvalues%,
      ByVal Guess#) As Double

   Function IRR (values() As Double, ByVal Guess As Double) As Double

      On Error GoTo IrrErr
      iArgMin% = LBound(values)
      cArg% = UBound(values) - iArgMin%
      IRR = IRRC#(values(iArgMin%), cArg%, Guess)
      Exit Function
IrrErr:
      MsgBox (Str$(Err))
      Exit Function

   End Function

   ' Enter the following Function statement on one, single line:
   Function MIRR (values() As Double, ByVal finance As Double,
      ByVal reinvest As Double) As Double

      On Error GoTo MirrErr
      iArgMin% = LBound(values)
      cArg% = UBound(values) - iArgMin%
      MIRR = MIRRC#(values(iArgMin%), cArg%, finance, reinvest)
      Exit Function
MirrErr:
      MsgBox (Str$(Err))
      Exit Function

   End Function

   Function NPV (ByVal Rate1 As Double, values() As Double) As Double

      On Error GoTo NpvErr
      iArgMin% = LBound(values)
      cArg% = UBound(values) - iArgMin%
      NPV = NPVC#(Rate1, values(iArgMin%), cArg%)
      Exit Function
NpvErr:
      MsgBox (Str$(Err))
      Exit Function

   End Function
```

3. Add the following code to your program in the Form_Click event:

```
   Sub Form_Click ()
      Static Values(5) As Double  ' Set up array.
      Fmt = "###,##0.00"  ' Define money format.
      Guess = .1  ' Guess starts at 10%.
      RetRate = .0625 ' Set fixed internal rate.
      Values(0) = -70000  ' Business start-up costs.
      ' Positive cash flows reflecting income for four successive years.
      Values(1) = 22000: Values(2) = 25000
      Values(3) = 28000: Values(4) = 31000
      NetPVal = NPV(RetRate, Values())    ' Calculate net present value.
      Msg = "The net present value of these cash flows is "
      Msg = Msg & Format(NetPVal, Fmt) & "."
      MsgBox Msg  ' Display net present value.
   End Sub
```

4. From the Run menu, choose start (ALT, R, S) or press the F5 key to run the program. You will see a message box that contains the correct Net Present Value result of 19,312.57.

Additional reference words: buglist3.00 3.00 errmsg
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOther

BUG: Incorrect Behavior in MaskedEdit BorderStyle Property
Article ID: Q101257

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic programming system
  for Windows, version 3.0

SYMPTOMS
========

Setting the BorderStyle property of the Masked Edit control to None at design time results in an "Invalid property value" error at run time. In addition, setting the Mask property to anything and then setting the BorderStyle property back to Single causes unusual characters to appear in the Mask property.

CAUSE
=====

The cause of the problem is unknown at this time.

WORKAROUND
==========

There is no known work around at this time.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File (ALT F, A), and add MSMASKED.VBX to your project.

3. Place a Masked Edit control (MaskedEdit1) on Form1.

4. Set the BorderStyle Property of MaskedEdit1 to 0 - None.

5. From the Run menu, choose start (ALT, R, S), or press the F5 key to run the program.

6. Visual Basic will generate an "Invalid Property Value" error. Click OK

in the error message to return to Visual Basic.

7. Set the Mask Property of MaskedEdit1 to #### and set the BorderStyle
   Property back to 1 - Single.

8. Now check the Mask Property. It contains unusual characters, but it
   should still contain ####.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

UPD: Invalid file format Error When Run VB app's EXE File
Article ID: Q101261

The information in this article applies to:

- The Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

You may encounter the following error when running a Visual Basic executable (EXE) file:

```
   Invalid file format
```

Or you may encounter the following error when loading a Visual Basic project or form:

```
   Error loading '<form filename>'. A control could not be loaded due
   to a load error. Continue?
```

CAUSE
=====

This problem will occur when you have installed a new version of a custom control and the internal property list of the control has incorrectly changed in a way that breaks backward compatibility.

This problem is known to occur when you have installed the Visual Basic version 3.0 GRID.VBX file over an earlier version of the grid. Specifically, the problem will occur for an existing Visual Basic application, built using a previous version of the grid, that sets the HelpContextID property of the grid.

In the case where the problem occurs when you load a project into Visual Basic that contains a grid, the problem will only occur when the form file(s) containing the grid have been saved in binary format.

This problem is also known to occur when using Visual Basic version 2.0 and the CMDIALOG.VBX control. For more information on this problem, please see the following article in the Microsoft Knowledge Base:

```
ARTICLE-ID: Q100611
TITLE     : FIX: VB 2.0 Prof Demo Causes Error: Invalid File Format
```

WORKAROUND
==========

There are several ways that you can work around this problem:

If you are using a Visual Basic version 3.0 application and you encounter this problem, you can:

- Acquire an updated copy of GRID.VBX from Microsoft (see instructions in the More Information section below).

 - Replace the Visual Basic version 3.0 of GRID.VBX with an earlier
   version. A disadvantage of this strategy is that applications requiring
   the Visual Basic version 3.0 grid will not run.

If you are a developer of a Visual Basic version 3.0 application that uses
the grid, you can:

 - Acquire an updated copy of GRID.VBX from Microsoft (see instructions in
   the More Information section below). You will need to build your
   application using this grid.

 - Rename GRID.VBX to a different name such as MSGRID3.VBX and rebuild the
   application using the renamed grid. A disadvantage of this strategy is
   that the grid will not be automatically updated when a new version of
   the grid (such as a version of the grid containing bug fixes) is
   released.

The following shows the date, time, size, and version number of the
GRID.VBX file that leads to this problem:

```
  Date: 28-APR-1993
  Time: 12:00 a.m.
  Size: 44667
  Version: Not Marked
```

The following shows the date, time, size, and version number of the
GRID.VBX file that fixes this problem:

```
  Date: 15-JUNE-1993
  Time: 5:26 p.m.
  Size: 45136
  Version: 03.00.0538
```

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. The problem is corrected by the updated
version of GRID.VBX.

MORE INFORMATION

How to Obtain Updated Copy of GRID.VBX

To obtain the updated copy of GRID.VBX, download VBGRID.EXE, a self-
extracting file, from the Microsoft Software Library (MSL) on the following
services:

 - CompuServe
      GO MSL
      Search for VBGRID.EXE
      Display results and download

 - Microsoft Download Service (MSDL)

      Dial (206) 936-6735 to connect to MSDL
      Download VBGRID.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get VBGRID.EXE

Steps to Reproduce Problem
----

1. Start 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 a Visual Basic version 1.0 or 2.0 version of GRID.VBX to Form1.

3. Put a grid control (Grid1) on Form1

4. Set the HelpContextID property of Grid1 to 1 (or some non-zero value).

5. From the File menu, choose Make EXE File (ALT, F, K) and create an EXE called PROJECT1.EXE.

6. Replace the older version of grid with the Visual Basic version 3.0 version of GRID.VBX, which has a date and time of 28-APR-1993 12:00 am.

7. Run the PROJECT1.EXE file created in step 5.

You should encounter an "Invalid file format" error. If you replace the Visual Basic version 3.0 grid with the version of the grid used in Step 2 and re-run PROJECT1.EXE, the program should run correctly.

Additional reference words: 3.00 softlib update3.00 S14643
KBCategory: kbprg kbfile kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Problems Printing Projects to HPLJ4

Article ID: Q101379

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, versions 2.00 and 3.00

SYMPTOMS
========

All the Form text and/or code will be printed together on a single page
if from the VB.EXE programming environment, you select one or more of
the combinations listed below from the Print dialog and print to an HP
LaserJet 4/4M printer using the HPLJ4 printer driver (the HPPCL5E file
version 31.V1.08).

Here are the problem combinations:

- Form & Form Text
- Form & Code
- Form & Form Text & Code

CAUSE
=====

This is caused by a bug in the HPLJ4 printer driver (HPPCL5E.DRV version
31.V1.08).

WORKAROUND
==========

There are two possible ways to work around this problem:

- Print each piece of the project separately. First print the
  Form, and then print the Form text and/or code.

- Use the HPLJIII printer driver (HPPCL5.DRV) with the HPLJ4 printer.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

This problem does not occur if you do not print the Form graphic. If
you print only the Form text and/or code, it will print as expected.

Selecting Current or All from the Print dialog does not effect the
problem.

When an updated driver is available that solves this problem we will
post that information here in the Microsoft Knowledge Base.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprint kbprg kbbuglist
KBSubcategory: APrgPrint

BUG: ALT+MINUS SIGN Does Not Work with Maximized MDI Forms
Article ID: Q101380

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, versions 2.00 and 3.00

SYMPTOMS
========

When you press the ALT+MINUS SIGN key combination in an application that has an MDI Form and MDI child form, the control box on the MDI child form should receive the focus and the system menu should drop down. But this does not happen if the MDI child form is maximized.

WORKAROUND
==========

Instead of using the ALT+MINUS SIGN key combination, use the following two steps to drop down the system menu for a maximized MDI child form:

1. Press the ALT Key to activate the control box for the maximized MDI child form
2. Press the ENTER key to drop down the system menu.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: GP Fault When Opening Menu Design Window in VB.EXE

Article ID: Q101381

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, version 3.0

SYMPTOMS

At times, you may receive a general protection (GP) fault when opening the Menu Design window in Visual Basic for Windows. This can result in the loss of all current additions and changes you made to your project since you last saved it.

CAUSE

This is caused by a bug in the VB.EXE environment where a pointer is referenced after being invalidated. In this case, it happens when you assign text to a Tag property for one of the menu items already on the form and you do not save your form immediately prior to opening the Menu Design Window.

WORKAROUND

To avoid this occasional GP fault, either do not set the Tag property of a menu item at design time or always save your work before opening the Menu Design Window.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist3.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: VB Dynasets Incorrectly Bypass Defaults on SQL Server

Article ID: Q101522

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When inserting a row into a SQL Data Source using dynasets, you may see one of the following behaviors:

 - The row is not inserted due to a NON-NULL integrity conflict.
 - The row is inserted but the default for a column is bypassed.

The behavior depends on the table definition (can it be made NULL or not) for the default-bound column. If default(s) exist on the table in SQL Server and the dynaset column corresponding to the default-bound column is not given a value before the insert, one the behaviors listed above will occur:

CAUSE
=====
On the Update method for the Dynaset, the following SQL code is generated by Jet Engine used by both Microsoft Access and Visual Basic version 3.0:

```
   Insert into Customer (Name, City) values ("bob", NULL)
```

For example, look at the schema definition shown in the More Information section below. If the table definition is as in A, the Insert fails because it is an attempt to insert NULL into a non-null column. If the table definition is as in B, the Insert command inserts "bob" and Null into the table -- bypassing the default of "Seattle" for City

To correct the problem, the Jet Engine should construct the SQL Statement to enforce defaults:

```
   Insert into Customer (Name) values ("bob")
```

This would correctly insert "bob" and "Seattle" into the Customer table.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Example to Reproduce Problem

The following example demonstrates this incorrect behavior:

```
// SQL Server schema definition

// A) City is defined 'non-nullable' for behavior (1) to manifest
   Create table Customer
   (Name char(30) not null , City char(30) not null)

// B) City is defined 'nullable' for behavior (2) to manifest
   Create table Customer
   (Name char(30) not null , City char(30) null)

   Create Unique Index Customer_ndx on Customer(name)
   Create Default city_default  as "Seattle"
   sp_bindefault city_default, 'table.city'

// VB Code to insert a new row into SQL Server
   Dim DS as Dynaset
   DS = DB.Createdynaset ("Customer")
   DS.AddNew
   DS("Name") = "bob"
// No code to set the value for 'City'
   DS.Update
   DS.Close
```

If the table definition for Customer is as in A, an attempt to
insert a new row into SQL Server fails with the following message
from SQL Server:

```
   Column 'Name' in table 'Customer' may not be NULL.
```

If the table definition for Customer is as in B, the row is
inserted into SQL Server, but the default has been bypassed. The
values "bob" and Null are inserted into the table

Additional reference words: buglist3.00 3.00 Access JET default update
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: APrgDataODBC

BUG: Bad Result If Multiple Aggregate Functions in SQL Stmt
Article ID: Q101553

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When an SQL query statement contains multiple aggregate functions, the result set incorrectly contains the same value for all the functions. The result of the first aggregate function is duplicated in the result column of all of the other functions.

CAUSE
=====
Aggregate functions typically do not contain explicit column names for expressions in the SQL query. In SQL queries containing multiple aggregate function calls, the Access database layer does not uniquely identify the return columns for any functions past the first. Therefore, it duplicates the result column of the first function in the result columns of the succeeding functions

This problem did not occur in Visual Basic version 2.0. In Visual Basic version 3.0, the Microsoft Access engine was integrated into the data access functionality. The Microsoft Access engine tracks the column by name, whereas Visual Basic version 2.0 tracks the column by the column offset.

WORKAROUND
==========

Use aliases for the aggregate functions to solve the problem. Replace the SQL statement shown below in the "Steps to Reproduce Problem" section with the following SQL statement, which contains the aliases One and Two for the column names for the separate SUM expressions:

```
   Select SUM(PubID) as One, SUM(Au_ID) as Two From Titles
```

The Alias names can be anything other than the column name and must be unique within the statement.

After inserting the aliases, run the SQL statement again and notice that the two fields now correctly show the different results.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic.

2. From the File menu, choose Open Project (ALT, F, O). Open VISDATA.MAK from the VB\SAMPLES\VISDATA directory.

3. From the Run menu, choose start (ALT, R, S) or press F5 to run the program.

4. From the Visual Data File menu, choose OpenDatabase. From the sub menu choose MS Access.

5. From the Open MS Access Database dialog box, select the BIBLIO.MDB file.

6. For the RecordSet Form Type, select Grid.

7. Enter the following SQL statement in the SQL Statement window:

   ```
   Select SUM(PubID), SUM(Au_ID) From Titles
   ```

8. Click the Execute SQL command Button.

9. The result shows in a grid window. The two fields have the same value. They should be different.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: Out of Memory w/ Var Named ClientLeft/Top/Width/Height
Article ID: Q102069

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When you use a variable named ClientLeft, ClientTop, ClientWidth, or ClientHeight without explicitly defining the variable with Dim or Global, Visual Basic incorrectly generates the error "Out of memory - insufficient variable space," error code 3761.

WORKAROUND
==========

Define the variable using Dim or Global. For example:

```
   Dim ClientLeft As Single
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem

```
Sub Form_Click ()
    ' any of the following statements cause the error
    Print ClientLeft
    Print ClientTop
    Print ClientWidth
    Print ClientHeight
End Sub
```

Additional reference words: buglist3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOther

BUG: Setup Wizard Error: Sharing Violation Reading Drive C:
Article ID: Q102478

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, version 3.0

SYMPTOMS
========

The error message "Sharing Violation on drive C:" is displayed during the compression stage when using the Setup Wizard tool included with Visual Basic version 3.0 for Windows.

CAUSE
=====

This is caused by the combination of the file sharing utility SHARE.EXE, the compression utility COMPRESS.EXE, and the Setup Wizard tool SETUPWIZ.EXE. The problem occurs when the compression utility tries to open the files SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, or CMDIALOG.VBX.

This problem does not occur when running under Windows for Workgroups version 3.1 in Enhanced mode, because Windows for Workgroups version 3.1 does not use the file sharing utility SHARE.EXE. It uses its own file sharing utility (VSHARE.386).

WORKAROUND
==========

If you need to use the file sharing utility SHARE.EXE, copy SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, and CMDIALOG.VBX from the \WINDOWS\SYSTEM directory to the directory where the SETUPWIZ.EXE file is located. Then the SETUPWIZ.EXE and COMPRESS.EXE program will not try to use the same files at the same time. Set the Read-Only attribute of all four files, regardless of their actual location.

If you are running Windows in 386 Enhanced mode, you can also workaround the problem by using the VSHARE.386 virtual device driver rather then the MS-DOS SHARE.EXE program.

To obtain the VSHARE.386 driver, download WW1000.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
  GO MSL
  Search for WW1000.EXE
  Display results and download

- Microsoft Download Service (MSDL)
  Dial (206) 936-6735 to connect to MSDL
  Download WW1000.EXE

- Internet (anonymous FTP)

```
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get WW1000.EXE
```

If you are unable to access the sources listed above, you can have the driver mailed to you by calling Microsoft Product Support Services Monday through Friday, 6:00 A.M. to 6:00 P.M. Pacific time at (206) 637-7098. Ask for Application Note identification number WW1000.EXE. If you are outside the United States, contact the Microsoft subsidiary for your area. To locate your subsidiary, call Microsoft International Customer Service at (206) 936-8661.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist3.00 3.00
KBCategory: kbtool kbbuglist kbfile
KBSubcategory: TlsSetWiz

BUG: Domain Functions Available Only Within SQL Statement
Article ID: Q102479

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, version 3.0

SYMPTOMS
========

If you try to use the domain aggregate function examples provided in the Microsoft Visual Basic version 3.0 for Windows Help file, you will receive this error message:

```
   Reference to undefined Function or array
```

CAUSE
=====

The examples for the domain aggregate functions are incorrect. The domain aggregate functions, like the SQL aggregate functions, can be used only within an SQL statement.

WORKAROUND
==========

Use the domain aggregate functions within an SQL statement, as in the following example. Enter the following as one, single line:

```
   Set Dn = Db.CreateDynaset("Select DAvg(""AU_ID"", ""AUTHORS"")
      FROM Authors")
```

STATUS
======

Microsoft has confirmed this to be a bug in the Visual Basic version 3.0 Help file. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Step-by-Step Example

The following example demonstrates how to print to the form an average of all the AU_ID values in the Authors table from the BIBLIO.MDB database that comes with Microsoft Visual Basic version 3.0 for Windows:

1. Start Visual Basic or from the File menu, choose New Project if Visual Basic is already running. Form1 is created by default.

2. Add the following code to the Click event of Form1:

```
   Sub Form_Click()
      Dim Db As Database
      Dim Dn As Dynaset

      Set Db = OpenDatabase("C:\VB\BIBLIO.MDB")
      ' Note: enter the following Set Dn code as one, single line.
      Set Dn = Db.CreateDynaset("Select DAvg(""AU_ID"", ""AUTHORS"")
         FROM Authors")
         Print Dn(0) ' This is the equivelant of
                     ' Form1.Print Dn.Fields(0).Value
      ' It is always a good idea to close the database objects:
      Dn.Close
      Db.Close

   End Sub
```

3. Run the example. Then click the form.

All the other domain aggregate functions work in a similar way. It is only the example that is incorrect in the Visual Basic Help file. The other information explaining how to use the function parameters is correct.

The Following are the Domain Aggregate Functions:

```
   DAvg
   DCount
   DFirst
   DLast
   DLookup
   DMin
   DMax
   DStDev
   DStDevP
   DSum
   DVar
   DVarP
```

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: Can't Load Custom Control DLL: PICCLIP.VBX in Windows 3.0
Article ID: Q102649

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic programming system
  for Windows, version 3.0

SYMPTOMS
========

You receive the following error when you try to run the Professional
Edition of Microsoft Visual Basic version 3.0 for Windows using the
Microsoft Windows version 3.0 operating system.

```
   Can't load Custom Control DLL: 'C:\WINDOWS\SYSTEM\PICCLIP.VBX'
```

WORKAROUND
==========

Update your operating system to Microsoft Windows version 3.1, or edit
the AUTOLOAD.MAK file to delete the reference to the PICCLIP.VBX file.
Then restart Visual Basic for Windows.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft producsts listed
at the beginning of this article when used with the Microsoft Windows
version 3.0 operating system. We are researching this problem and will post
new information here in the Microsoft Knowledge Base as it becomes
available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic in Microsoft Windows version 3.0.

2. If you have not modified the AUTOLOAD.MAK, you will receive the error:

```
   Can't load Custom Control DLL: 'C:\WINDOWS\SYSTEM\PICCLIP.VBX'
```

Additional reference words: buglist3.00 3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Out of Memory w/ MSOLE2.VBX When SHARE.EXE Not Loaded
Article ID: Q103438

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0.

SYMPTOMS
========

You receive an "Out of Memory" error after adding the MSOLE2.VBX control to a form. That is, after adding the MSOLE2.VBX control to a form, you proceed to select an item from the list. Once you press the OK button, you get the "Out of Memory" error. This is an incorrect error message.

CAUSE
=====

This error can be caused by not having SHARE.EXE loaded in memory. The MSOLE2.VBX control requires that SHARE.EXE be loaded in memory before you use the MSOLE2.VBX control. The problem is that the error message is incorrect. You are not out of memory. Instead of "Out of Memory," the error message should say "SHARE.EXE required to perform this operation."

WORKAROUND
==========

Close Windows. Go to the \DOS directory and run SHARE.EXE to load it into memory. Then restart Windows and Visual Basic. Now you can add a MSOLE2.VBX control to your form, select an option from the list, and choose OK to see the desired embedded object appear on your form.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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 a MSOLE2.VBX control to Form1.

3. Once the control displays the Insert Object window, select an object from the list provided, and choose the OK button. This should result in the "Out of Memory" error.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: GPF in VB.EXE at 0038:3B6F w/ Compile-Time Error & Set
Article ID: Q105140

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When you try to run a program within the development environment, a general protection (GP) fault occurs immediately in module VB.EXE at 0038:3B6F.

CAUSE
=====

The problem can occur when there is a compile-time error (such as a syntax error) followed by a Set statement where the left hand side of the Set is not a simple object variable. The compile-time error does not have to involve an object variable. Examples of object variables that are not simple are object arrays and nested OLE objects.

```
   Static a(10) As Form
   Set a(i) = Form1      ' setting an object array element

   Static b As Object
   Set b = CreateObject(...)
   Set b.c = ...          ' setting an object variable within an object
```

WORKAROUND
==========

Find and correct the compile-time error. This takes some effort because the GP fault occurs before VB.EXE shows the location of the error. To narrow down the search for the statement causing the error, remove Set statements from your code until the GP fault no longer occurs. Then correct all compile-time errors, and put the Set statements back in.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The following code reproduces the problem:

```
   Sub Form_Load ()
      Static a(0) As Form
      Print 1 >= "a"        ' type-mismatch error
      Set a(0) = Nothing
   End Sub
```

Additional reference words: buglist3.00 3.00 UAE GPF
KBCategory: kbenv kbbuglist
KBSubCategory: EnvtDes

BUG: Overflow in VB version 3.0 ICONWRKS Sample Program
Article ID: Q105808

```
----------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.00
----------------------------------------------------------------------
```

SYMPTOMS
========

The ICONWRKS (Icon Works) sample program that shipped with Visual Basic version 3.0 can fail with an "Overflow" error when you attempt to do a File+Open from the Editor form, on some high-resolution monitors. ICONWRKS is installed by default under the subdirectory \SAMPLES\ICONWRKS.

CAUSE
=====

ICONWRKS fails with "Overflow" in the Extract_Image_And_Mask procedure in ICONWRKS.BAS on the following line:

```
R = SetBitmapBits(editor.Pic_Image.Image, ImageSize, Lpicon + 12 + 128)
```

The statement DEFINT A-Z at the top of the module makes the variable R an integer. However, the API function SetBitmapBits returns a Long Integer when run on some high-resolution monitors. NOTE: This problem may also occur on other lines with other API calls.

This sample program was developed under Visual Basic version 1.0 and was not updated for 3.0.

RESOLUTION
==========

To correct the problem, add the following statement to ICONWRKS.GBL:

```
   Global R As Long
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. This problem is corrected as described in the Resolution section of this article.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOther

BUG: VB Printer.Width/Height Values Incorrect for Plotter
Article ID: Q106495

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

If you set up the HP Draftmaster II plotter with a paper size of A0 in the Windows Control Panel, Visual Basic returns incorrect values for the Printer.Width and Printer.Height properties. If you set the plotter's paper size to A3 or A4, then Printer.Width and Printer.Height return correct values.

CAUSE
=====

The Printer.Width and Printer.Height properties are designed to receive an integer only. Plotter paper sizes often exceed an integer. This causes an overflow in the Width and Height properties.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Choose the Printers dialog from the Windows Control Panel.

2. Select the "HP Draftmaster II (HP Plotter)" and click Add. Click Install to add this printer to the Installed Printers list.

   NOTE: Windows for WorkGroups installs the necessary files in the WINDOWS\SYSTEM directory.

3. Make "HP Draftmaster II (HP Plotter)" the default printer.

4. Click Setup. Select the options DEVICE.DRAFTMASTER II and SIZE.A0. Click OK.

5. Click Close to close the Printers dialog.

6. Start Visual Basic.

7. Press the F5 key followed by CTRL+BREAK.

8. Activate the Debug window and execute the following statements:

```
   Debug.Print Printer.Width
   Debug.Print Printer.Height
```

The problem is that Printer.Width and Printer.Height return incorrect values such as 161.

Additional reference words: buglist3.00 3.00 Hewlett-Packard H-P
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOther

BUG: VB Setup Files Modified or Corrupted, Using \WINDOWS Path
Article ID: Q106496

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When you run SETUP.EXE to install Visual Basic version 3.0, you may receive the following error message:

```
   Setup Files Have Been Modified or Corrupted.
```

This message is misleading and incorrect.

CAUSE
=====

This behavior will occur if your PATH statement contains

```
   \WINDOWS
```

If you modify the path to read C:\WINDOWS, SETUP.EXE works correctly. You can confirm your current PATH by running the PATH command at the MS-DOS prompt.

This problem occurs in Visual Basic version 3.0 SETUP, but does not occur in SETUP for earlier versions.

RESOLUTION
==========

Here are several ideas to help you solve this problem. If one doesn't work, then move on to the next item in the list:

1. Modify the PATH statement in your AUTOEXEC.BAT file to be C:\WINDOWS instead of \WINDOWS.

2. Check for viruses.

3. Try it on a different computer.

4. Close all your applications, and temporarily disable the Startup group in Windows. In Program Manager, click the Startup group. Then choose Properties from the File menu, and change the name to Xstartup. Then turn the computer off and on in order to clear memory.

5. Copy the files to a \VBSETUP directory, for example:

```
   Disk #1 -->  \VBSETUP\DISK1
   Disk #2 -->  \VBSETUP\DISK2
   Disk #3 -->  \VBSETUP\DISK3
```

   Then run setup from the DISK1 directory.

   NOTE: This takes quite a bit of space on the hard drive, so it is a
   last resort.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed
at the beginning of this article. We are researching this problem and will
post new information here in the Microsoft Knowledge Base as it becomes
available.

Additional reference words: buglist3.00 3.00
KBCategory: kbsetup kbbuglist
KBSubcategory: SetIns

BUG: Name Not Found in This Collection When Deleting Member
Article ID: Q107362

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  version 3.0

SYMPTOMS
========

The Delete method incorrectly reports the following message for the existing member under certain conditions:

```
   Name not found in this collection. Error 3265.
```

This occurs when you use the OpenDatabase function to open a database, and then immediately, as the first change to the database's structure, execute a Delete method on a member of a TableDefs or Indexes collection. The member can be a TableDef or Index.

CAUSE
=====

The problem occurs when a Delete method is the first data definition language (DDL) operation after you open the database.

WORKAROUND
==========

To work around the bug, use the Refresh method on the Indexes collection before using the Delete method. An example is shown in "Workaround Example" under the More Information section below.

NOTE: A program will correctly give the above error message when the name truly is not found in the collection. As soon as a Delete method succeeds on a specified TableDef or Index member of a collection, that name will no longer be found in the collection.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Copy the file BIBLIO.MDB from your Visual Basic directory to the root directory (C:\BIBLIO.MDB). The program created below modifies the copy of BIBLIO.MDB instead of the master file. The BIBLIO.MDB sample database file is a bibliographical reference.

3. Add a command button to Form1.

4. Add the following code to the Command1 Click event:

```
   Sub Command1_Click()
      Dim db as Database
      Set db = OpenDatabase ("c:\BIBLIO.MDB")
      ' db.TableDefs("Titles").Indexes.Refresh ' Add this for workaround
      db.Tabledefs("Titles").Indexes.Delete "PubID"  '<- Problem line
   End Sub
```

5. Start the program or press the F5 key. The program gives the incorrect error, "Name not found in this collection."

Workaround Example

To work around this bug, use the Refresh method before using the Delete method:

```
   db.TableDefs("Titles").Indexes.Refresh
```

As an alternative workaround, replace the Delete line with this command:

```
   ' Enter the following two lines as one, single line:
   db.TableDefs("Titles").Indexes.Delete
      db.TableDefs("Titles").Indexes("PubID")
```

This command expands "PubID" into its complete reference:

```
   db.TableDefs("Titles").Indexes("PubID")
```

This refreshes the Indexes collection before PubID is deleted in the same statement.

NOTE: If you run the program twice using the workaround, the program correctly gives the error, "Name not found in this collection." The error is correct this time because the PubID index member was successfully deleted and no longer exists.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataOther

BUG: Incorrect VB Error When Delete Index on Open Table
Article ID: Q107363

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

If you attempt to delete an index on an open table, you correctly get an error but the message is incorrect.

The program example given in the More Information section gives the following incorrect error when attempting to delete an index from an open Microsoft Access table:

```
   ODBC-call failed.
```

This message is misleading because the program uses no ODBC. This is error number 3146, returned by the Err function.

CAUSE
=====

The ODBC-call failed message is incorrect. The message should instead say the table is currently open and cannot be locked.

You cannot delete an index from a table if the table is Open. This is behavior is by design. You must be able to lock the table before you can delete an index. You cannot lock the table if the table is open by anyone.

WORKAROUND
==========

Close the table before deleting an index. You may also need to use the Refresh method on the TableDefs collection before using the Delete method.

STATUS
======

Regarding the incorrect error message, Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

All other behavior described in this article is by design.

MORE INFORMATION
================

Steps to Reproduce Behavior
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following code to the Form Load event:

```
Sub Form_Load ()

   Const DB_LANG_GENERAL = ";LANGID=0x0809;CP=1252;COUNTRY=0"
   Dim db As database
   If Dir$("c:\t.mdb") <> "" Then Kill "c:\t.mdb"
   Set db = CreateDatabase("c:\t.mdb", DB_LANG_GENERAL)

   Dim f1 As New field
   Dim f2 As New field
   f1.Name = "field1"
   f1.Type = 3  ' integer
   f2.Name = "field2"
   f2.Type = 3  ' integer

   Dim td As New TableDef
   td.Name = "table1"
   td.Fields.Append f1
   td.Fields.Append f2

   Dim ix As New Index
   ix.Name = "index1"
   ix.Fields = "field1;field2"
   td.Indexes.Append ix

   ' create the table
   db.TableDefs.Append td

   ' add records to the table
   Dim tb As table
   Set tb = db.OpenTable("table1")
   tb.AddNew
   tb.Fields("field1").Value = 1
   tb.Fields("field2").Value = 2
   tb.Update
   tb.AddNew
   tb.Fields("field1").Value = 4
   tb.Fields("field2").Value = 5
   tb.Update
   tb.AddNew
   tb.Fields("field1").Value = 7
   tb.Fields("field2").Value = 8
   tb.Update

   tb.Index = "index1"
   tb.Seek "=", 4, 5
   Print tb.NoMatch
   Print tb.Fields("field1").Value

   ' Delete the index:
   Dim td2 As TableDef
   Set td2 = db.TableDefs("table1")
```

```
      ' The following line causes "ODBC-call failed" error message:
      td2.Indexes.Delete db.TableDefs("table1").Indexes("Index1").Name
      ' The workaround is to move this statement to after the table Close

      tb.Close
      ' Workaround: move the statement from above to here:
      ' td2.Indexes.Delete db.TableDefs("table1").Indexes("Index1").Name
      db.Close

   End Sub
```

3. Start the program or press the F5 key.

This program gives the incorrect error message "ODBC-call failed", err=3146, when attempting to delete an index from the Access database. This message is misleading because the program uses no ODBC.

To work around the problem, close the table before doing the Delete method.

NOTE: If the first data definition language (DDL) operation is a Delete method, the Delete will fail with the error, "Name not found in this collection." This is a separate bug and is explained in another article in the Microsoft Knowledge Base. To work around this bug, execute the db.TableDefs.Refresh method before attempting a Delete.

Additional reference words: buglist3.00 3.00 erase remove how-to create
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataOther

BUG: First Item Can Disappear in Outline Control Style 0 or 2

Article ID: Q108659

The information in this article applies to:

 - Professional Edition of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS

When an Outline custom control has both:

 - Style property value of 0 or 2
 - Indent property value of 0 on an item

the first visible item in the outline incorrectly disappears when you initially click any other item at run time. The problem occurs both in the Visual Basic environment and in compiled EXE files.

By design, the item that has an Indent property of 0 should not display. However this should not have any effect on the items that do display.

WORKAROUND

The first item reappears as soon as you select it with the mouse or keyboard. The keyboard interface for the Outline control includes LEFT ARROW, RIGHT ARROW, UP ARROW, DOWN ARROW, HOME, END, PAGE UP, PAGE DOWN, plus sign (+), and minus sign (-).

You can prevent the disappearance of the first item as follows:

 - Do not use an Indent value of 0 on items in an Outline control that uses Style property values of 0 or 2. Instead, use an Indent value of 1 or greater.

or

 - Add the Outline1.ListIndex=1 statement after you add all items and indents to the Outline control. This ListIndex method selects the first item automatically, working around the problem.

STATUS

Microsoft has confirmed this to be a bug in the Professional Edition of Microsoft Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

NOTE: An item that has an Indent property of 0 will be visible when you use Style property values of 1, 3, 4, and 5, which include pictures or tree lines. The bug mentioned in this article does not occur for these styles.

Steps to Reproduce Behavior
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. From the File menu, choose Add File. Add the MSOUTLIN.VBX control file
   from your WINDOWS\SYSTEM directory.

3. Add an Outline custom control to the form.

4. Select the Outline control and press the F4 key to display the
   Properties window. Set the Style property to 0 or 2:

```
      0 - Text Only
   or
      2 - Plus/Minus and Text
```

5. Double-click the form to display the code window. Add the following code
   to the Form Load event:

```
   Sub Form_Load ()
      For i = 0 To 4
         ' Note that item 0 will not be visible, by design.
         outline1.AddItem Str$(i)
         outline1.Indent(i) = i
      Next
      For i = 1 To 4
         outline1.Expand(i - 1) = True
      Next
      ' Add the following statement to work around the bug:
      ' Outline1.ListIndex=1
   End Sub
```

6. Start the program, or press the F5 key. To duplicate the problem, click
   any item except the first. The first item, 1, incorrectly disappears.

   As long as you click any item except the first, the first item remains
   invisible. As soon as you click the first item, it correctly appears.

To work around the problem, add Outline1.ListIndex=1 to the end of the code listed in step 5. The ListIndex method selects the first item automatically.

REFERENCES
==========

 - "Microsoft Visual Basic Version 3.0: Professional Features Book 1:
   Custom Control Reference." See the Outline control, pages 256-257.

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Out of Memory Error When Adding 35-50 Pen Controls
Article ID: Q110989

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, versions 2.0 and 3.0

SYMPTOMS
========

You will receive an "Out of Memory" error message if you attempt to add more than approximately 36 HEdit or approximately 50 BEdit Pen Controls to an individual Form.

WORKAROUND
==========

You can edit in only one control at a time, so use only one of each of the Pen Controls on top of multiple Labels or Picture Box Controls. This will display the result of the edit when you move the focus to another control.

The examples in the More Information section below will demonstrate how to do this for each of the Pen Controls.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Here are two examples showing how to work around this bug:

Step-by-Step Example One

The following example demonstrates how to use multiple Picture Boxes and one HEdit Control with the DelayRecog = True. This will allow the user to input signatures into a field which will then remain as written, no recognition is performed.

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following controls, and set the indicated properties:

```
   Control           Property     Value
   ----------------------------------------
   Picture Box       Name         PictureB
                     Index        0
```

```
                  AutoRedraw   True

   Picture Box      Name         PictureB
                    Index        1
                    AutoRedraw   True

   HEdit Control    Name         HEdit1
                    DelayRecog   True
                    Text         <blank>

   Command Button   Name         Command1
                    Caption      Clear
```

3. Add the following code to the General Declarations section of Form1:

```
   'Enter each of the following Declare statements on one, single line:
   Declare Function GetDC Lib "USER" (ByVal hWnd As Integer) As Integer
   Declare Function BitBlt Lib "GDI" (ByVal hDestDC As Integer,
      ByVal x As Integer, ByVal y As Integer, ByVal nWidth As Integer,
      ByVal nHeight As Integer, ByVal hSrsDC As Integer,
      ByVal xSrc As Integer, ByVal ySrc As Integer, ByVal dwRop) As Integer

      Const SRCCOPY &H00CC0020&   ' Will be used in the call to BitBlt
      Dim LastPos As Integer      ' Will be used to keep track of the
                                  ' last edited field.
      Dim InkArray(2) As String   ' This array will be used to store the
                                  ' InkDataString for each of the fields.
```

4. Add the following code to the appropriate Control's event procedures:

```
   Sub Command1_Click ()
      HEdit1.InkDataString = ""     ' Clear the field.
   End Sub

   Sub Form_Load ()
      Call PictureB_Click(0)    ' Position the HEdit control over the
   End Sub                      ' first field.

   Sub Picture1_Click (index As Integer)
      ' Copy the image in the HEdit control to the Picture Box
      destDC = GetDC(HEdit.hWnd)
      ' Enter the following three lines as one, single line:
      dummy% = BitBlt(PictureB(LastPos).HDC, 0, 0,
         PictureB(LastPos).ScaleWidth-2, PictureB(LastPos).ScaleHeight-2,
         destDC, 1, 1)
      InkArray(LastPos) = HEdit1.InkDataString
      ' Save the Ink data for this field.  It will be reassigned
      ' to the HEdit control the next time that field is selected
      ' and the HEdit control is positioned on top of it.
      LastPos = index        ' Update LastPos to the current field.
      HEdit1.Visible = False ' This prevent a flicker when the
                             ' control is moved.
      HEdit1.Top = PictureB(index).Top     ' Move the HEdit Control on
      HEdit1.Left = PictureB(index).Left   ' top of the selected Picture
      HEdit1.Width = PictureB(index).Width ' box field.
      HEdit1.Height = PictureB(index).Height
      HEdit1.InkDataString = InkArray(index) ' Reset the Ink in the
```

```
                                             ' Hedit Control to that
                                             ' stored for this field.
      HEdit1.Visible = True
   End Sub
```

Step-by-Step Example Two

The following example demonstrates how to use multiple Labels and one HEdit Control with DelayRecog = False. This works using a BEdit control as well. This allows the user to input hand-written characters into a field where those characters will then be recognized and converted into font characters.

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following controls, and set the indicated properties:

Control	Property	Value
Label	Name	LabelF
	Index	0
	Caption	<blank>
Label	Name	LabelF
	Index	1
	Caption	<blank>
HEdit Control	Name	HEdit1
	DelayRecog	False
	Text	<blank>

3. Add the following code to the General Declarations section of Form1:

```
   Dim LastPos As Integer
```

4. Add the following code to the appropriate event procedures:

```
   Sub Form_Load ()
      Call LabelF_Click (0)
   End Sub

   Sub LabelF_Click (index As Integer)
      LabelF(LastPos).Caption = HEdit.Text    ' Copy the contents of
                                              ' the HEdit Control to  the
                                              ' Label Control.
      LastPos = index         ' Update LastPos to the current field.
      HEdit1.Visible = False  ' This prevent a flicker when the
                              ' control is moved.
      HEdit1.Top = LabelF(index).Top       ' Move the HEdit Control on
      HEdit1.Left = LabelF(index).Left     ' top of the selected Picture
      HEdit1.Width = LabelF(index).Width   ' Box field.
      HEdit1.Height = LabelF(index).Height
      HEdit1.Text = LabelF(index).Caption  ' Reset the Ink in the HEdit
                                           ' Control to that stored for
                                           ' this field.
      HEdit1.Visible = True
   End Sub
```

Additional reference words: buglist2.00 buglist3.00 2.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

Buglist for Visual Basic 3.0 for Windows as of 26-Apr-1994

Article ID: Q111476

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0

SUMMARY

=======

This single article lists the unfixed bugs in Visual Basic version 3.0 for Windows. Each of these bugs is completely described in an article in the Microsoft Knowledge Base. The article identification numbers (Q numbers) and titles are listed below. For more information on any of these bugs, please see the complete article in the Microsoft Knowledge Base.

If you prefer, you can have any of these articles faxed to you by using Microsoft FastTips. To use this service, call (800) 936-4300 and follow the prompts. When prompted to enter the Item ID, enter the Q number (without the Q) to have the bug article faxed to you.

A similar article lists the fixed bugs. To get both these lists, query on the following word in the Microsoft Knowledge Base:

   kblist

===

UNFIXED BUGS IN VISUAL Basic VERSION 3.0 FOR WINDOWS

===

ARTICLE-ID	TITLE
Q73700	BUG: TABs Paste Incorrectly as | to VB.EXE's Immediate Window
Q73839	BUG: Scroll Box Flashing Not Updated If Bar Resized w/ Focus
Q74132	BUG: [Character May Sort Incorrectly in List or Combo Box
Q74194	BUG: Can Click in Code Window Without Activating it in VB.EXE
Q74409	BUG: Pressing ESC or CTRL+BREAK Makes Mouse Pointer Disappear
Q74564	BUG: Incorrectly Accessing System Menu of Hidden Form
Q75640	BUG: ExtFloodFill Won't Fill Over QBColors If AutoRedraw=True
Q76514	BUG: Duplicate Procedure Name Alters Original Capitalization
Q76520	BUG: No Option Button Active (Dotted) in Frame
Q76555	BUG: Italic and Large Fonts Display Poorly in Text Boxes
Q76628	BUG: Dir List Box Does Not Give Error 68 Device Unavailable

Q76983	BUG: FormName Not in Correct Order After Out of Memory Error
Q77393	BUG: DateSerial Does Not Give Error for Invalid Month or Day
Q77734	BUG: Incorrect Focus Shift for Disabled Control in Break Mode
Q77738	BUG: Extra Click Event if Double-Click When Mouse Button Down
Q77928	BUG: CTRL+LEFT/RIGHT ARROW Behaves Differently When Edit/Type
Q78380	BUG: Using Nonstandard Icons Can Cause UAE/GP Fault/Hang
Q78892	BUG: Multiline Text Box Contents Not Gray When Enabled=False
Q79240	BUG: Invalid outside Sub Error When Copy or Paste to General
Q79241	BUG: Resetting ListIndex Property Generates Click Event
Q79242	BUG: Some Property Values May Be Incorrect in Maximized Form
Q79602	BUG: Option Button w/ Focus Selected When Click Form Caption
Q80023	BUG: Click Event May Fail to Occur in Cascading Menu
Q80286	BUG: TAB Character Can Incorrectly Cause KeyUp/KeyDown Events
Q80780	BUG: No Resources Causes Failed to Open Graphics Server Error
Q80905	BUG: Gauge Custom Control: No Error for Illegal NeedleWidth
Q80967	BUG: Grid Custom Control: Scroll Bars Displayed Unnecessarily
Q81187	BUG: Gauge Custom Control: Valid NeedleWidth Range 1 to 32767
Q81449	BUG: 3-D Panel Control Doesn't Resize to Key Status Control
Q81460	BUG: Vertical Linear Gauge Loses Upper Border's Bottom Pixels
Q81461	BUG: InnerBottom/InnerRight Defines Gauge Fill Area Badly
Q81472	BUG: Graph: ExtraData May Not Say: Invalid Property Value
Q81951	BUG: 3D Command Button Shows Outline when Outline = False
Q81955	BUG: Scroll Control: UAE/GPF If Drag Method in GotFocus Event
Q81998	BUG: Grid: No Error Changing FixedAlignment on Non-Fixed Col
Q83463	BUG: Graph Axis Titles Don't Switch on Horizontal Bar Graphs
Q84236	BUG: VB Graph Custom Control: SeeThru Paints Incorrectly
Q84269	BUG: Must Call API to Print Color Text on Color Printer in VB
Q84553	BUG: THREED.VBX: Command/Group Push Buttons Show Invalid File
Q94778	BUG: Illegal function call / Division By Zero Errors

Q95197	BUG: Stack Fault When Move Sets Tiny Width in 2-Item Combo Box
Q95430	BUG: GPF/UAE If Multi-Select Controls w/ No Common Properties
Q95431	BUG: Type Mismatch Error If Use VAL Function on Big Hex Value
Q95499	BUG: Stack Fault May Occur If Trapping Divide By Zero
Q95500	BUG: GPF When Close Form That Contains a Single MCI Control
Q95513	BUG: Neg ScaleHeight Resizes Control When Form Saved as ASCII
Q95830	BUG: Stack Fault When Move Makes Combo Box Width Too Small
Q99705	BUG: Changing Default Printer Doesn't Effect Printer.Fonts
Q99872	BUG: Wrong Menu Click Event After Hiding Menu
Q99873	BUG: MaskedEdit MaxLength Reset to 64 When Mask=""
Q100190	BUG: Overflow Error When CurrentX Or CurrentY Greater Than 32K
Q100191	BUG: VB Pro Setup Fails to Correctly Associate .HLP Files
Q100192	BUG: Out of Memory Error on Show Next from Debug Menu
Q100193	BUG: 3D Button Loses 256-Color Palette When Load 2nd Bitmap
Q100195	BUG: Grid Control Repaints When Another Form Is Made Active
Q100327	BUG: Unload in 3D GroupPush Button Causes GP Fault
Q100367	BUG: Referencing Data Object Gives Error: Object not an Array
Q100610	BUG: GPF in Some Video Drivers When Load RLE Bitmaps > 20K
Q100612	BUG: Font3D Property Set Incorrectly in THREED.VBX Controls
Q100613	BUG: Data Access Setup Can Give Incorrect Error Message
Q101245	BUG: Ref to NPV / IRR / MIRR Gives Undefined Functions Error
Q101257	BUG: Incorrect Behavior in MaskedEdit BorderStyle Property
Q101379	BUG: Problems Printing Projects to HPLJ4
Q101380	BUG: ALT+MINUS SIGN Does Not Work with Maximized MDI Forms
Q101381	BUG: GP Fault When Opening Menu Design Window in VB.EXE
Q101522	BUG: VB Dynasets Incorrectly Bypass Defaults on SQL Server
Q101553	BUG: Bad Result If Multiple Aggregate Functions in SQL Stmt
Q102069	BUG: Out of Memory w/ Var Named ClientLeft/Top/Width/Height

Q102478	BUG: Setup Wizard Error: Sharing Violation Reading Drive C:
Q102479	BUG: Domain Functions Available Only Within SQL Statement
Q102649	BUG: Can't Load Custom Control DLL: PICCLIP.VBX in Windows 3.0
Q103438	BUG: Out of Memory w/ MSOLE2.VBX When SHARE.EXE Not Loaded
Q103976	BUG: Invalid Argument Err on Execute Method w/ SQL Passthrough
Q105140	BUG: GPF in VB.EXE at 0038:3B6F w/ Compile-Time Error & Set
Q105171	BUG: Error 13 (Type Mismatch) & Error 3061 w/ SQL Queries
Q105808	BUG: Overflow in VB version 3.0 ICONWRKS Sample Program
Q106495	BUG: VB Printer.Width/Height Values Incorrect for Plotter
Q106496	BUG: VB Setup Files Modified or Corrupted, Using \WINDOWS Path
Q107362	BUG: Name Not Found in This Collection When Deleting Member
Q107363	BUG: Incorrect VB Error When Delete Index on Open Table
Q108659	BUG: First Item Can Disappear in Outline Control Style 0 or 2
Q110989	BUG: Out of Memory Error When Adding 35-50 Pen Controls
Q113031	BUG: ActiveControl Property of Screen Object Loses Memory
Q113281	BUG: Long Field Names May Cause GP Fault in VB.EXE
Q113330	BUG: Edit Replace All Has Different Limit Than Specified
Q113331	BUG: Bound 3D Panel Control Won't Update When Caption Changed
Q113332	BUG: AddNew Method Gives Error: Illegal Function Call
Q113437	BUG: SQL Server GetDate() Function Error: Record is deleted
Q113438	BUG: Multiple CreateObject May Cause GP Fault in VBOA300.DLL
Q113681	BUG: Mouse Input Ignored When Display Modal Form in Spin Event
Q113686	BUG: Lost MouseDown Event with Command Button & Check Box
Q113687	BUG: Field Name Same as Reserved Word Can Cause GP Fault
Q113896	BUG: No Update When Delete All MESSAGE_SHOWADBOOK Recipients
Q113900	BUG: dBASE & FoxPro Memos Corrupted During Concurrent Addnew
Q114000	BUG: Nesting OLE Automation Calls Causes GP Fault

Additional reference words: buglist 3.00 kblist
KBCategory: kbref kbbuglist
KBSubcategory: RefsProd

BUG: Serial Port Driver for WFW 3.11 Sends Extra Byte

Article ID: Q112418

```
---------------------------------------------------------------------
The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0
---------------------------------------------------------------------

SYMPTOMS
========

The MSCOMM.VBX custom control may appear to send an unexpected byte
when the port is closed by setting the PortOpen property to false.

CAUSE
=====

There is a known problem with the miniport driver SERIAL.386 that was
released with Windows for Workgroups 3.11. This is not a problem with
the MSCOMM control since it can be reproduced by calling the CloseComm
Windows API function directly.

WORKAROUND
==========

To get an updated version of SERIAL.386, download WG1001.EXE, a
self-extracting file, from the Microsoft Software Library (MSL)
on the following services:

 - CompuServe
      GO MSL
      Search for WG1001.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download WG1001.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get WG1001.EXE

STATUS
======

Microsoft has confirmed this to be a problem in Windows for Workgroups
version 3.11. We are researching this problem and will post new
information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
--------------------------
```

1. Connect two machines using a null modem cable. You must run Windows for Workgroups 3.11 on the machine running Visual Basic.

2. Start a new project in Visual Basic, Form1 is created by default.

3. Add an MSCOMM (Comm1) control and a command button (Command1) to Form1.

4. Add the following code to the Command1 Click event procedure:

```
   Sub Command1_Click ()
      comm1.PortOpen = True
      comm1.PortOpen = False
   End Sub
```

5. Start the Terminal application on the second machine. In Terminal choose Settings Communications (ALT, S, C) and change the Baud Rate to 9600.

6. From the Run menu in Visual Basic on the first machine, choose Start (ALT, R, S) or press the F5 key to run the program. Click the Command1 button and the machine running Terminal will indicate that a byte has been transmitted from closing the port.

The problem can also be reproduced with the following method:

1. Connect two machines using a null modem cable. You must run Windows for Workgroups 3.11 on the machine running Visual Basic.

2. Start a new project in Visual Basic. Form1 is created by default.

3. Add a command button (Command1) to Form1.

4. Add the following Declare statements to the General declarations section of Form1:

```
   ' Enter each of the following Declare statements on one, single line:
   Declare Function OpenComm Lib "User" (ByVal lpComName As String, ByVal
      wInQueue As Integer, ByVal wOutQueue As Integer) As Integer
   Declare Function CloseComm Lib "User" (ByVal nCid As Integer)
      As Integer
```

5. Add the following code to the Command1 Click event procedure:

```
   Sub Command1_Click ()
      Dim id As Integer, success As Integer
      id = OpenComm("COM1", 1024, 128)
      success = CloseComm(id)
   End Sub
```

6. Start the Terminal application on the second machine. In Terminal choose Settings Communications (Alt, S, C) and change the Baud Rate to 9600.

7. From the Run menu in Visual Basic on the first machine, choose Start (ALT, R, S) or press the F5 key to run the program. Click the Command1 button and the machine running Terminal will indicate that a byte has been transmitted from closing the port.

Additional reference words: serial comm port
KBCategory: kbprg kbcode kbbuglist kbfile
KBSubcategory: PrgCtrlsCus

UPD: SQORA.DLL Does Not Allow Lengthy SQL Statements
Article ID: Q112446

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0
- Microsoft Access, version 1.1

SYMPTOMS
========

If the table and field names are long or the query is complex, executing a query or updating a record in an Oracle table results in the following error message:

```
   Statement was longer then allowable maximum 2000+ chars
```

CAUSE
=====

This occurs because of a problem with SQORA.DLL, the Oracle ODBC driver.

RESOLUTION
==========

Obtain and install the updated driver (instructions are provided in the More Information section below), or use queries to do updates rather than updating records with the Oracle table in Datasheet view. The query should yield only the columns to be updated. For complex queries, reduce both the number of tables or joins in the query and the number of fields used or shown in the query. This reduces the lengths of SQL statements.

STATUS
======

Microsoft has confirmed this to be a problem in the Oracle ODBC driver shipped with Microsoft Access version 1.1 and the Professional Edition of Microsoft Visual Basic version 3.0. An updated driver that corrects this specific problem is available for owners of Microsoft Access version 1.1 or the Professional Edition of Microsoft Visual Basic version 3.0.

MORE INFORMATION
================

How to Obtain the Updated Driver

The updated Oracle ODBC driver (SQORA.DLL) is available for use by registered owners of:

- Microsoft Access version 1.1
- Professional Edition of Visual Basic version 3.0

By downloading the new driver, you are indicating that you own one or both of these two products. To obtain the updated driver, download and then run

ORA110.EXE, a self-extracting file.

Download ORA110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for ORA110.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download ORA110.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get ORA110.EXE
```

Contents of ORA110.EXE

README.TXT - a copy of this article
ORACLE.TXT
SQORA.DL_
SQORASTP.DL_
ODBC.INF
SETUP.EXE

NOTE: The SETUP.EXE file is called by the ODBC control panel facility and will not run as a stand-alone file.

How to Install the Updated Driver
----

1. Start Windows if it is not running.

    - If you are running Windows 3.1, open Control Panel.
    - If you are running Windows 3.0 or NT, select the ODBC program group.

2. Double-click the ODBC icon.

3. From the Data Sources dialog, select the Drivers... button.

4. From the Drivers dialog, select Add...

5. Enter the drive letter and directory from which you are installing.

6. Select Oracle from the list of available drivers, and choose OK. ODBC setup will install the driver at this point. If an ODBC Oracle driver of the same version number or higher exists on the hard disk, ODBC setup will ask if you want to replace it. In most cases, you will want to stay with the most recent version.

7. Choose close, and you are finished.

What ODBC Setup Installed

The ODBC installation installed a new SQORA.DLL, a new SQORASTP.DLL, and a new ORACLE.TXT to your Window's system directory.

		Old		New
SQORA.DLL	Version:	1.00.2816	Version:	1.00.3112
	Size:	143,600 bytes	Size:	144,096 bytes
	Date:	4/16/93	Date:	7/12/93
SQORASTP.DLL	Version:	1.00.2403	Version:	1.00.3106
	Size:	9,328 bytes	Size:	9,632 bytes
	Date:	5/7/93	Date:	7/6/93

Oracle drivers are manufactured by Oracle Corporation and Btrieve drivers by Novell, Inc. These two vendors are independent of Microsoft; we make no warranty, implied or otherwise, regarding these products' performance or reliability.

Additional reference words: 1.10 3.00 update3.00 softlib S14637
KBCategory: kbinterop kbfile kbbuglist kbtool
KBSubCategory: RefsProd

BUG: ActiveControl Property of Screen Object Loses Memory

Article ID: Q113031

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

It is possible to receive an "Out of Memory" error when using the
ActiveControl property of the Visual Basic Screen object.

CAUSE
=====

The ActiveControl property of the screen object can leak memory when used
with the Is operator.

RESOLUTION
==========

Use a temporary variable to work around the problem. For example, change
the command click event in the code listed in the More Information section
below to this code:

```
Sub Command1_Click ()
   Dim ctr As Long
   Dim ActControl As control
   Do
      ctr = ctr + 1
      Text1.Text = ctr
      ' Use a temporary object variable
      Set ActControl = Screen.ActiveControl
      If ActControl Is Command1 Then
      End If
   Loop
End Sub
```

This code should run indefinitely.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0
for Windows. We are researching this problem and will post new information
here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Put a command button (Command1) and a text box (Text1) on the form.

3. Add the following code to the command button's click event:

```
   Sub Command1_Click ()
      Dim ctr As Long
      Do
         ctr = ctr + 1
         Text1.Text = ctr
         If Screen.ActiveControl Is Command1 Then
         End If
      Loop
   End Sub
```

4. Run the program, and you will receive an "Out of Memory" error
   eventually.

Additional reference words: buglist3.00 3.00 MemLeak
KBCategory: kbprg kbbuglist
KBSubcategory: PrgOptMemMgt

BUG: Edit Replace All Has Different Limit Than Specified
Article ID: Q113330

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

Performing an Edit Replace All on a long line of code in the Visual Basic Environment may generate this error message:

   One or more replacements too long and not made.

CAUSE
=====

The Visual Basic for Windows environment limits the length of individual lines of code to 1023 characters. No line longer than this may be input. However, if you choose Edit Replace from the Visual Basic menu with the Replace All button selected, the environment erroneously limits the total line length to 254 characters.

WORKAROUND
==========

To create a line longer than 254 characters, do it in separate lines and then concatenate the lines together by deleting the carriage return between the lines. A better solution would be to break up the single long line into multiple shorter lines. For additional information, please see the following article in the Microsoft Knowledge Base:

```
ARTICLE-ID: Q94696
TITLE     : How to Break Long Statements into Multiple Lines
```

STATUS
======

Microsoft has confirmed this to be a bug in the products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following code to the Form_Load event:

```
   Sub Form_Load ()
```

```
      Dim S As String
      ' Enter the following four lines as one, single line:
      S = "This is a test.This is a test.This is a test.This is a test.
      This is a test.This is a test.This is a test.This is a test.
      This is a test.This is a test.This is a test.This is a test.
      This is a test.This is a test.This is a test.This is a test.This xx"
      ' The previous string's quoted text is 247 characters in length.
      ' The total line length is 254 characters.
   End Sub
```

3. Choose replace-all from the edit menu, and choose to change 'xx' to 'xxx' to generate the following error message:

   ```
   One or more replacements too long and not made.
   ```

   The help text on this error message indicates that the replace operation has reached a 1023 character maximum:

   "A replace operation has attempted to create a line of code longer than the 1023-character maximum allowed. Those replacements where the resulting line is less than 1023 characters were completed."

   This is not true. The line length has not exceeded 1023 characters.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: AddNew Method Gives Error: Illegal Function Call

Article ID: Q113332

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0
---------------------------------------------------------------------
```

SYMPTOMS
========

It is possible to receive an "Illegal Function Call" error message in response to an AddNew method. This article demonstrates how this error can occur and how to work around it.

CAUSE
=====

The cause of this problem has not yet been determined. We're researching it.

WORKAROUND
==========

There are two ways to work around this problem. The first method is to move to the last record before adding the record. The following code demonstrates this work around:

```
   Form1.AutoRedraw = True
   Data1.DatabaseName = "DB.MDB"
   Data1.RecordSource = "SELECT * FROM Table1 WHERE Field1 = 'Record1'"
   Data1.Refresh
   Data1.Recordset.MoveLast
   Print "ReadOnly "; Data1.ReadOnly
   Print "Options "; Data1.Options
   Print "Updatable "; Data1.Recordset.Updatable
   Data1.Recordset.AddNew
```

The other method is to delete the index. In the previous example, you would delete Index1.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

A problem with the Microsoft Access database engine can cause an "Illegal Function Call" error when adding a record to a database. This error should not occur. This problem is not related to the data control and can also occur when using methods from the Professional Edition of Visual Basic version 3.0.

Steps to Reproduce Problem
----

1. Use the Data Manager to create a database with these characteristics:

```
   DATABASE: DB.MDB
   TABLES:   Table1
   FIELDS:   Field1 in Table1
   INDEXES:  Index1 on Field1 not Unique and not Primary
```

2. Add two records to Table1 with values of "Record1" and "Record2" for Field1.

3. Start a new project in Visual Basic. Form1 is created by default.

4. Add a data control (Date1) and a command button (Command1) to the form.

5. Add the following code to the command button's click event:

```
   Sub Command1_Click ()
      Form1.AutoRedraw = True
      Data1.DatabaseName = "DB.MDB"
      Data1.RecordSource = "SELECT * FROM Table1 WHERE Field1 = 'Record1'"
      Data1.Refresh
      Print "ReadOnly "; Data1.ReadOnly
      Print "Options "; Data1.Options
      Print "Updatable "; Data1.Recordset.Updatable
      Data1.Recordset.AddNew
   End Sub
```

6. Run the code and click the Command1 button. You should receive the "Illegal function call" error on the on the Addnew line.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: SQL Server GetDate() Function Error: Record is Deleted

Article ID: Q113437

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  version 3.0

SYMPTOMS
========

Error 3167 "Record is deleted" can occur as records from the Dynaset are fetched when both of the following conditions are present:

- The Dynaset is opened on a table in a SQL Server database that has an index based on a field of type date/time.

- That indexed date/time field has been populated by the native SQL server function GetDate().

The records on which the error will occur are not predictable but are consistent.

WORKAROUND
==========

Any one of the following three possible workarounds will work:

- Use a snapshot object instead, or use a Dynaset with the DB_SQLPASSTHROUGH option, which is functionally the same thing as a snapshot.

- Drop the index before and rebuild it after Visual Basic does the update if a non-passthrough Dynaset is needed in order to update the table in question.

- Let Visual Basic do the updates to that table using the Now function instead of the stored procedure.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

If records are added directly by Visual Basic (filling the date/time field by using the Now function), there is no problem. But if the indexed date/time field is assigned by the GetDate() function, the error occurs.

Using GetDate() and a Stored Procedure that May Result in the Problem
----

The GetDate() function is an intrinsic function native to Microsoft SQL Server. The most likely situation is for this to be called from within a stored procedure, which in turn is called from a Visual Basic program. The following is the SQL Server syntax used to create a stored procedure that adds records to a SQL Server table named GetDateBugTab:

```
create proc getdatebug
As
declare @dt datetime,
@messagestr varchar(39)
select @dt=GetDate()
select @messagestr = "This record added from stored proc"
insert into GetDateBugTab
(fDateTime, fsourceofdate)
select @dt,@messagestr
```

The structure of GetDateBugTab is reported by executing the system procedure sp_help GetDateBugTab against the SQL Server database.

Results look somewhat like this:

```
Name                           Owner                        Type
GetDateBugTab                  dbo                          user table
Column_name   Type             Length Nulls Default_name    Rule_name
------------- ---------------- ------ ----- --------------- ---------------
fDateTime     datetime         8       1     (null)          (null)
fsourceofdate varchar          39      1     (null)          (null)
index_name    index_description                             index_keys
}ndx          nonclustered, unique located on default       fDateTime
```

The stored procedure would be executed from Visual Basic by using code such as this:

```
Dim db As database
' Enter the following two lines as one, single line of code:
Set db = OpenDatabase("", 0, 0,
   "odbc;uid=sa;pwd=;DSN=sqlserver2;database=playpen2;")
label1 = db.ExecuteSQL("getdatebug")
```

Filling the Table by Using Visual Basic Directly Causes No Problem
----

By contrast, if the table is filled by Visual Basic code, no problem occurs. For example, the following code works without a problem:

```
Dim db As database
' Enter the following two lines as one, single line of code:
Set db = OpenDatabase("", 0, 0,
   "odbc;uid=sa;pwd=;DSN=sqlserver2;database=playpen2;")
dt$ = Now
' Enter the following two lines as one, single line of code:
label1 = db.ExecuteSQL("insert into GetDateBugTab (fDateTime,fsourceofdate)
   select '" & dt$ & "', 'This is from VB Now function'")
```

Using SQLPASSTHROUGH Still Causes a Problem

Alternatively, the entire body of the stored procedure can be sent to the SQL Server from Visual Basic. This is because the ExecuteSQL uses the SQLPASSTHROUGH flag and sends the syntax to the SQL Server for processing. This will still cause the error, however.

```
Dim db As database
' Enter the following two lines as one, single line of code:
Set db = OpenDatabase("", 0, 0,
   "odbc;uid=sa;pwd=;DSN=sqlserver2;database=playpen2;")
dt$ = Now
' Enter the following four lines as one, single line of code:
label1 = db.ExecuteSQL("declare @dt datetime, @messagestr varchar(39)select
   @dt=GetDate() select @messagestr = "This record added from stored proc"
   insert into GetDateBugTab (fDateTime,fsourceofdate)
   select @dt,@messagestr")
```

REFERENCES
==========

More information about calling stored procedures is documented in the following Microsoft SQL manual which covers the Visual Basic Library for SQL Server: "Microsoft SQL Server Programmer's Reference for Visual Basic."

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataOther

BUG: Multiple CreateObject May Cause GP Fault in VBOA300.DLL
Article ID: Q113438

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

A general protection (GP) fault may result when working with OLE objects in Visual Basic if you repeatedly create new OLE objects.

CAUSE
=====

When OLE objects are created with Visual Basic and that OLE object is subsequently set to Nothing, a hidden instance of the OLE application is spawned and then orphaned. This uses up system resources and eventually either the machine will hang (stop responding to input) or a GP fault will occur in VBOA300.DLL at 0001:0D03.

WORKAROUND
==========

When you create OLE objects. Be sure to close or quit the OLE object before setting the variable to Nothing. Please see the example at the end of this article.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a command button (Command1) to the form.

3. Add the following code to the Command1_Click event:

```
   Sub Command1_Click ()
      Dim X As object
      Dim R As object
      Dim Iterations As Integer

      While True
```

```
         Iterations = Iterations + 1
         Debug.Print Iterations
         Set X = CreateObject("Excel.Sheet.5")

         ' Enter the following two lines as one, single line:
         Set R = X.Parent.Sheets(1).Range(X.Parent.Sheets(1).Cells(2, 2),
            X.Parent.Sheets(1).Cells(52, 2))

         Set R = Nothing
         Set X = Nothing
      Wend
   End Sub
```

4. Run the program.

When this code is run, the program will eventually produce a GP fault.
Closing the WorkBook will not circumvent this problem. You must quit the
application to avoid the GP fault.

Example Workaround

The following code will not produce a GP fault:

```
   Sub Command1_Click ()
      Dim X As object
      Dim R As object
      Dim Iterations As Integer

      While True
         Iterations = Iterations + 1
         Debug.Print Iterations
         Set X = CreateObject("Excel.Sheet.5")

         ' Enter the following two lines as one, single line:
         Set R = X.Parent.Sheets(1).Range(X.Parent.Sheets(1).Cells(2, 2),
            X.Parent.Sheets(1).Cells(52, 2))

         ' The next line quits the application for an Excel object
         X.Application.Quit
         Set R = Nothing
         Set X = Nothing
      Wend
   End Sub
```

Additional reference words: buglist3.00 GPF EXCEL 5.00 WINWORD 6.00 VBASIC
3.00
buglist3.00
KBCategory: kbole kbbuglist
KBSubCategory: IAPOLE

BUG: Mouse Input Ignored When Display Modal Form in Spin Event

Article ID: Q113681

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0

SYMPTOMS
========

While displaying a modal form in the SpinDown() or SpinUp() event of the SPIN.VBX custom control, your computer ignores all mouse actions along with any system key requests such as CTRL+ESC and ALT+TAB. However, normal keyboard input is unaffected. In other words, you can still press TAB to move between controls, press the ENTER key on a command button, and enter text in a text box.

WORKAROUND
==========

When running in the Visual Basic environment, you can press CTRL+BREAK to get mouse control and system keyboard input back.

To work around this bug, enable a timer in the SpinDown() or SpinUp() event, and display your modal form in the Timer() event.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

It is important to note that a MsgBox statement, which is application modal, operates correctly. The problem only appears when displaying a modal form.

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. From the File menu, choose Add File. In the Files box, select the SPIN.VBX custom control file. The Spin control appears in the Toolbox.

3. Add a spin control (Spin1) to Form1.

4. Add the following code to the SpinDown() event of the Spin1 control:

```
   Sub Spin1_SpinDown ()
```

```
      form2.Show 1
   End Sub
```

5. From the File menu, choose New Form (ALT, F, F). Form2 is created by default.

6. Add a text box (Text1) and a command button (Command1) to Form2.

7. Add the following code to the Click() event of Command1:

```
   Sub Command1_Click ()
      Unload Me
   End Sub
```

8. Press the F5 key to run the program. Then click the Down arrow of the spin control. Form2 is displayed. Now if you try to click the command button or do anything with the mouse, your input is ignored. Also, if you try to pop up the windows task list by pressing CTRL+ESC, your input is ignored.

9. Enter text in the text box, and press the TAB key to move the focus to the command button. Then press the ENTER key to invoke the Command1 Click event. Form2 is unloaded and the mouse input is restored to normal.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00 buglist2.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Lost MouseDown Event with Command Button & Check Box
Article ID: Q113686

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0

SYMPTOMS
========

If a user repeatedly clicks a command button or check box, the control does not receive all the MouseDown events, but it does receive all the MouseUp events.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

The following steps demonstrate that a MouseDown event is lost after repeatedly clicking a command button.

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a Command button (Command1) to Form1.

3. Add the following code to the Command1_MouseDown event procedure of Form1:

```
   Sub Command1_MouseDown ()
      Static i
      i = i + 1
      Debug.Print "MouseDown - "; i
   End Sub
```

4. Add the following code to the Command1_MouseUp event procedure of Form1:

```
   Sub Command1_MouseUp ()
      Static j
      j = j + 1
      Debug.Print "MouseUp - "; j
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run

   the program.

6. Click the Command1 button repeatedly. The debug window will show
   that fewer MouseDown events are generated than MouseUp events.

Additional reference words: buglist2.00 buglist3.00 2.00 3.00 buglist2.00
buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Field Name Same as Reserved Word Can Cause GP Fault

Article ID: Q113687

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

It is possible to receive a general protection (GP) fault in VB.EXE at the address 0025:3182 when using the data access features of Visual Basic. You may also notice corruption in your source code if the source is saved in the binary format and reloaded. The problem may not occur all the time.

CAUSE
=====

This problem stems from using field names that are reserved words in Visual Basic. The problem occurs when using the following syntax to reference the field:

```
   Recordset![Reserved]
```

where "Recordset" is a table, dynaset, snapshot, or data control recordset, and "Reserved" is any Visual Basic reserved word.

RESOLUTION
==========

To avoid the problem use the Fields collection to refer to the field. For example:

```
   Recordset.Fields("Reserved").Value
   Recordset("Reserved")
```

where "Recordset" is a table, dynaset, snapshot, or data control recordset, and "Reserved" is any Visual Basic reserved word.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 will be created by default.

2. Add the following code to the Form Load event:

```
   Sub Form_Load()
```

```
      Dim tb As Table
      tb![To] = 12
   End Sub
```

3. Run the program. You should get an error and the code will be changed to the following.

```
   Sub Form_Load()
      Dim tb As Table
      tb!To = 12
   End Sub
```

Additional reference words: buglist3.00 3.00 buglist3.00 GPF
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: No Update When Delete All MESSAGE_SHOWADBOOK Recipients

Article ID: Q113896

```
---------------------------------------------------------------------
The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 3.0
---------------------------------------------------------------------
```

SYMPTOMS
========

When using the MAPI message action MESSAGE_SHOWADBOOK (11) while composing a message, you can select entries from the address book and those entries are reflected in the recipients list. But if you then go into the address book and delete all the entries, then when the dialog box terminates, it will not update the recipients list. It leaves all prior entries unmodified.

As a result, there's no way to detect if the user has used the dialog box to delete all the entries on the "To:" line.

WORKAROUND
==========

If you use MESSAGE_SENDDLG to bring up the MAPI compose dialog box, you can click the address button to display the address book. If you delete all the entries from the dialog here, the changes are reflected in the Send dialog box. The problem only occurs when you bring up the Address book dialog box directly.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add two text boxes (Text1, Text2), a command button (Command1), a MAPISession control (MAPISession1), and a MAPIMessages control (MAPIMessages1) to Form1.

3. Add the following constants to the general declarations section:

```
   Const MESSAGE_SHOWADBOOK = 11
   Const SESSION_SIGNON = 1
```

4. Add the following code to your form in the appropriate event procedures:

```
   Sub Form_Load ()
      MAPISession1.Action = SESSION_SIGNON
      MAPIMessages1.SessionID = MAPISession1.SessionID
   End Sub

   Sub Command1_Click ()
      MAPIMessages1.MsgIndex = -1
      MAPIMessages1.AddressEditFieldCount = 1
      MAPIMessages1.Action = MESSAGE_SHOWADBOOK
      Text1.Text = Str(MAPIMessages1.RecipCount)
      Text2.Text = ""
      For i = 0 To MAPIMessages1.RecipCount - 1
         MAPIMessages1.RecipIndex = i
         Text2.Text = Text2.Text & MAPIMessages1.RecipDisplayName & ";"
      Next i
      MAPIMessages1.RecipIndex = 0
   End Sub
```

5. Press the F5 key to run the program.

6. Click the command button and select some entries in the address book. You will see them in the text box.

7. Now click the button again. Delete all the entries in the dialog box. When you click OK and return, you'll see that it is as if nothing happened in the dialog box. This diminishes the usability of the Address dialog box.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgOther

BUG: dBASE & FoxPro Memos Corrupted During Concurrent Addnew

Article ID: Q113900

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0
- Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer

SYMPTOMS
=======

When two or more programs concurrently edit and/or add new records to the same dBASE or FoxPro table and that table contains a memo field, the memo field can become corrupted.

CAUSE
=====

This is not a bug Visual Basic itself, but a bug in the Microsoft Jet Database Engine used by Visual Basic.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

This problem arises when two or more Edit or Addnew operations are pending on the same table at the same time. When the record is updated, whichever program updates the memo field last has its memo data written into the memo field of the previously updated record.

Steps to Reproduce Problem

To duplicate this problem, you must already have a dBASE or FoxPro table that has a memo field in it.

1. From Visual Basic's Window menu, choose the Data Manager (ALT, W, A).

2. Restore the Data Manager's window to normal by clicking the Restore/Max button (ALT, SPACE, R).

3. Open a dBASE or FoxPro database that contains a table with a memo field (ALT, F, O).

4. Select the table from the Tables list, and choose Open.

5. Repeat steps 1 through 4 to provide a second instance of the Data

   Manager editing your table.

6. For the first instance, add a new record and edit the data in the memo
   field to say Instance1.

7. For the second instance, add a new record and edit the data in the memo
   field to say Instance2.

8. Choose Update and Save the Changes for the first instance. Repeat for
   the second.

9. Choose Refresh to update the recordset. When you view the records,
   you'll see that the memo fields in both instances read Instance2.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbinterop kbprg kbbuglist
KBSubcategory: APrgDataIISAM

BUG: Nesting OLE Automation Calls Causes GP Fault
Article ID: Q114000

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

Nesting multiple OLE Automation property accesses and/or method invocations within the same procedure may lead to temporary memory loss or a general protection (GP) fault.

CAUSE
=====

When Visual Basic makes the cross process calls to perform the property access or method invocation it does not release the temporary space allocated until the procedure completes execution.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Multiple IDispatch Calls Use Up Memory

In order to access a property or execute a method on a OLE Automation object, automation controllers (such as Visual Basic) make IDispatch calls to the server application. Each time a IDispatch call is made, Visual Basic creates a temporary variable referencing the object. Visual Basic version 3.0 does not release the memory for these temporary variables until the procedure in which the property access or method invocation resides completes execution.

Therefore, if you have multiple property accesses or method invocations within the same procedure, the amount of free memory steadily decreases until the procedure completes execution. Nesting multiple OLE Automation statements that require IDispatch calls within the same procedure can produce a GP fault in module VBOA300.DLL at 0001:0D03.

It could be useful to estimate the number of IDispatch calls Visual Basic makes. The number of IDispatch calls Visual Basic executes for an individual statement is directly related to the number of properties or methods that are combined by using the dot notation to perform the statement. Thus a good rule of thumb is to count the number of "dots" in the statement. For example, the following statement makes two IDispatch calls:

```
   oXLSheet.Range("A1").Value = "Price"
```

Depending on how you access a collection, there will also be implicit IDispatch calls. For example, if you access a collection without the Item method, it is implicitly called.

The following statement makes five IDispatch calls - three explicit "dots" and two implicit calls to the Item method:

```
   oXLApp.Workbooks(1).Sheets(1).Range("A1").Value = "Price"
```

Steps to Reproduce Problem
----

1. Start Microsoft Excel. Then Start a new project in Visual Basic. Form1 is created by default.

2. Add a command button (Command1) to Form1.

3. Add the following code to the Command1_Click event procedure:

```
   Sub Command1_Click ()
      Dim ExcelSheet As Object
      Dim StartCell As Object
      Dim EndCell As Object
      Dim I as Integer
      Set ExcelSheet = CreateObject("excel.sheet.5")
      ' The following For Next loop makes 4000 IDispatch calls:
      For i = 1 to 1000
         Set StartCell=ExcelSheet.Cells(i,1)
         Set EndCell=ExcelSheet.Cells(i,5)
         ExcelSheet.Range(StartCell,EndCell).FormulaArray = "=3"
      Next
      ExcelSheet.Application.Quit
      Set ExcelSheet = Nothing
   End Sub
```

4. Press the F5 key to run program. Then click the command button.

An Excel Worksheet is created and data is inserted into the Range of cells A1:E1000 via OLE automation. The For..Next loop makes 4000 cross-process calls from Visual Basic to Excel to insert the data into the Range of cells. Depending on the amount of Virtual Memory available to Windows on the system, the above code can lead to a low memory state or cause a GP fault in module VBOA300.DLL at 0001:0D03.

Additional reference words: buglist3.00 3.00 GPF MemLeak buglist3.00
W_VBApp
KBCategory: kbole kbbuglist
KBSubcategory: IAPOLE

BUG: ALT+TAB Hangs MDI App with Activated OLE 2.0 Server

Article ID: Q114346

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

The ALT+TAB keystroke combination may cause the computer to hang (stop responding to input) when an MDI application is an activated OLE 2.0 server in the MSOLE2.VBX control. It may also cause a general protection (GP) fault.

CAUSE
=====

This problem is caused by MSOLE2.VBX when interpreting the menu negotiation protocol of some server applications.

WORKAROUND
==========

The only way to work around this is to use the OLE control on a non-MDI form.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a new MDI form (MDIForm1).

3. Make Form1 an MDI Child by setting the MDIChild property to true.

4. Add an OLE2 control to Form1. Embed a new Word version 6.0 Document
   when the insert object dialog comes up.

5. Make MDIForm1 the start up form. From the Options menu, choose
   Project... and then Start Up Form. Select MDIForm1.

6. Add the following code to the MDIForm1 Load event:

```
   MDIForm1.Show
   Form1.Show
   Form1!Ole1.Action = 7
```

7. Save the project.

8. Make an EXE from the project.

9. Run the EXE with all other applications shut down.

10. Press ALT+TAB. The computer will lock up (hang).

Additional reference words: buglist3.00 freeze switch task GPF 3.00
buglist3.00
KBCategory: kbole kbbuglist
KBSubcategory: IAPOLE

BUG: Disk or Network Error with Data Access Objects
Article ID: Q114771

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

Error 3043 (Disk or network error) can result when you run multiple instances of one Visual Basic program or you run multiple programs, and all instances or programs perform data access on the same database.

If the program does not explicitly close all the data access objects (by using db.close, for example), then the error will occur when the second instance of the program tries to work with a data access object.

WORKAROUND
==========

Close all data access objects (tables, dynasets, snapshots, and databases) explicitly. For example, if your program has any of the following statements:

```
   Dim db As database
   Dim ds As dynaset
   Dim sn As snapshot
   Dim tb As table

   Set db = OpenDatabase("<some database file>")
   Set ds = db.CreateDynaset("<some query>")
   Set sn = db.CreateSnapshot("<some query>")
   Set tb = db.OpenTable("<some table name>")
```

Execute the following close statements before the program ends:

```
   tb.close
   ds.close
   sn.close
   db.close
```

NOTE: If you place the .Close methods in the Unload or QueryUnload events, make sure you invoke these events before your program ends by using the Unload statement (for example, Unload Me). Be careful when using the End statement; it does not invoke the Unload or QueryUnload events.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start Visual Basic and open the VISDATA.MAK project located in the \VB\SAMPLES\VISDATA directory (ALT, F, O).

2. From the File menu, choose Make EXE File to create an executable.

3. Exit Visual Basic, and run File Manager.

4. Change directories to the \VB\SAMPLES\VISDATA directory and double-click VISDATA.EXE to run the program. Repeat this step so that two instances of VISDATA are running.

5. In both instances of VISDATA, open the BIBILIO.MDB database located in your Visual Basic directory \VB.

6. Close one instance of VISDATA.EXE.

7. Start another instance of VISDATA.EXE and try to open the BIBLIO.MDB database again. You will get Error 3043 - "Disk or Network Error."

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: SetupKit: Fuzzy Title Display in Setup Program
Article ID: Q114773

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

The title on the background of the SETUP1.EXE is displayed incorrectly.

CAUSE
=====

The ShowMainForm subroutine in SETUP1.FRM of the SETUP1.MAK project incorrectly displays the form and prints a title on the background of the form before it changes the scalemode of the form. Therefore, when the form is refreshed again, the Print statement prints in a slightly different location, giving the "fuzzy" look.

WORKAROUND
==========

To work around the problem, rearrange the order of displaying the form and changing the scalemode of the form by using these steps:

1. Start Visual Basic and open the SETUP1.MAK project located in the \VB\SETUPKIT\SETUP1 directory (ALT, F, O).

2. View the code in the SETUP1.FRM form, and go to the ShowMainForm subroutine.

3. The incorrect code should read as follows:

```
Sub ShowMainForm (Caption$)
   Screen.MousePointer = 11
   Setup1.Caption = Caption$
   Setup1.Move 0, 0, Screen.Width, Screen.Height * .85
   Setup1.Show
   Setup1.Refresh

   Setup1.ScaleMode = 2
   Setup1.FontSize = 24
   Setup1.FontBold = True
   Setup1.FontItalic = True

   DrawBackground
End Sub
```

   To correct the problem, move the Setup1.Show and Setup1.Refresh statements to the end of the subroutine. Also, delete the call to DrawBackground. The corrected code should read:

```
   Sub ShowMainForm (Caption$)
      Screen.MousePointer = 11
      Setup1.Caption = Caption$
      Setup1.Move 0, 0, Screen.Width, Screen.Height * .85

      Setup1.ScaleMode = 2
      Setup1.FontSize = 24
      Setup1.FontBold = True
      Setup1.FontItalic = True

      Setup1.Show
      Setup1.Refresh
   End Sub
```

4. From the File menu, choose Save File to save the changes to the Form. Because the SetupWizard uses the project SETUP1.MAK as its template when creating your distribution diskettes, the change will correct problems for any new setup diskettes.

5. Press the F5 key to run the program. You should see a clear display of "Loan Application Setup" on the background of the form. Click Exit Setup to exit out of the application.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: buglist3.00 3.00 buglist3.00 setup wizard setupwizard
KBCategory: kbtool kbbuglist
KBSubcategory: TlsSetWiz

BUG: Compacted 1.1 DB Becomes 2.0 DB w/ Compatibility Layer

Article ID: Q115779

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0
- Microsoft Access 2.0/Visual Basic 3.0 Compatibility Layer also called Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer

----

SYMPTOMS
========

Error 3041 "Incompatible database version" is the result when you use the Microsoft Access 2.0/Visual Basic 3.0 Compatibility Layer to open a Microsoft Access version 1.1 database that was compacted by using the CompactDatabase statement with the DB_VERSION10 option.

CAUSE
=====

According to the README.TXT file for the Compatibility Layer, after installing the Compatibility Layer, the CompactDatabase statement in Visual Basic version 3.0, when used with the DB_VERSION10 constant (from the file DATACONS.TXT), is supposed to generate a Microsoft Access version 1.1 database. The following table shows which type of database is supposed to be created when you use CompactDatabase with and without DB_VERSION10:

Jet Database Engine	Creates by Default	With DB_VERSION10
Version 1.1	Version 1.1 database	Version 1.0 database
Version 2.0	Version 2.0 database	Version 1.1 database

However, this is not what happens. There is a bug in the Compatibility Layer that causes the CompactDatabase statement to always create a Microsoft Access version 2.0 database regardless of the existence of the constant DB_VERSION10 as an option. Both the Office Development Kit (ODK) and Access Developer's Toolkit (ADT) versions of the Compatibility Layer have this problem.

WORKAROUND
==========

Use Microsoft Access version 1.1 or 2.0 to compact a version 1.1 database instead of using the CompactDatabase statement from the Compatibility Layer.

NOTE: Once a database is in Microsoft Access version 2.0 format, you cannot convert it back to Microsoft Access version 1.1 format directly. If you used CompactDatabase and ended up turning a version 1.1 database into a version 2.0 database, you need to restore the version 1.1 database from a backup copy, or rebuild it by extracting the information from the newly compacted version 2.0 database and placing it into a new version 1.1 database.

WARNING: If the Compatibility Layer has been installed and you're using a Microsoft Access version 1.1 database with a Visual Basic application, NEVER use CompactDatabase on the version 1.1 database if you want it to remain in version 1.1 format. Instead, use Microsoft Access itself to compact version 1.1 databases.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

The following steps reproduce the problem if the Compatibility Layer is installed on your computer:

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a command button (Command1) to Form1.

3. Place the following code in the Command1 Click event procedure:

```
   Sub Command1_Click ()
      Const DB_VERSION10 = 1               'Microsoft Access Version 1.1
      Const DB_LANG_GENERAL = ";LANGID=0x0809;CP=1252;COUNTRY=0"

      ' The following should compact a 1.1 Database into a 1.1 Database.
      ' But it actually compacts a 1.1 Database into a 2.0 Database.
      ' Enter the following code as one, single line:
      CompactDatabase "C:\VB\BIBLIO.MDB", "C:\VB\NEWBIB.MDB",
         DB_LANG_GENERAL, DB_VERSION10
   End Sub
```

4. Press the F5 key to run the program, and click the Command1 button. This results in an "Incompatible Database Version" error if the resulting compacted database, which is now in Microsoft Access version 2.0 format, is used on a computer that does not have the Compatibility Layer installed.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: Help Compiler Indents Text Laid Out Above Bulleted Text
Article ID: Q116030

```
----------------------------------------------------------------------
The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 3.0
- Help Compiler version 3.10.505
----------------------------------------------------------------------
```

SYMPTOMS
========

If you have bulleted text in your help file, the Help Compiler indents the paragraph preceding the paragraph that contains the bulleted text as if both paragraphs were bulleted. This problem occurs only with version 3.10.505 of the Help Compiler.

CAUSE
=====

Version 3.10.505 of the Help Compiler does not compile the Help file correctly.

WORKAROUND
==========

To work around the problem, edit the .RTF source file using Microsoft Word for Windows or another .RTF file editor. Use the indent markers on the ruler to change the indention of the blank line immediately preceding the bulleted text. This causes the indention for the blank line to be modified instead of the indention for the previous paragraph.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Microsoft Word for Windows and open the ICONWRKS.RTF file located in \VB\HC.

2. Below the line that reads "To learn how to use Help, press F1 or choose Using Help from the Help menu," add a new paragraph that extends across more than one line. Then press the ENTER key twice, and enter bulleted text.

3. Save the file, and go to the MS-DOS prompt. Type the following to compile the Help file:

```
HC ICONWRKS.HPJ
```

4. Double-click the resulting Help file in File Manager to open it. You will see that the formatting for the bulleted text affected the preceding paragraph.

5. To solve the problem, reopen the ICONWRKS.RTF file in Microsoft Word. Adjust the indention for the blank line immediately preceding the bulleted text. Compile the modified .RTF file. Then double-click it in File Manager to view it. You will see that the alignment now looks correct.

Additional reference words: buglist3.00 3.00 buglist3.00
KBCategory: kbtool kbbuglist
KBSubcategory: TlsHC

BUG: Incorrect Popup Menu Events Fired with Invisible Menus

Article ID: Q116058

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

When the following conditions are met, Visual Basic may incorrectly fire a menu's Click event for the first visible menu item under the first menu on the form:

- You use the PopupMenu method to display a menu that has its Visible property set to False.

- The PopupMenu is not the first (leftmost) menu on the form, and the first menu on the form also has its Visible property set to False.

Under these conditions, the Click event for the first visible menu item on the first menu will be fired immediately after the PopupMenu method is called, so the desired PopupMenu will not be displayed.

WORKAROUND
==========

To work around this problem, either:

- Make the first menu on the form visible.

Or:

- Place all popup menus on a separate form. Then you can leave all the menus on your popup menus visible, but make the form that contains them invisible. To make the form invisible, set the form's visible property to false. You can invoke popup menus from other forms by referring to the form name and then the menu name, as in this example:

```
PopupMenu Form2.Menu2
```

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following menu items:

Caption	Name	Visible
Menu1	mMenu1	False
....Menu1Item1	mMenu11	True
Menu2	mMenu2	False
....Menu2Item1	mMenu21	True

3. Add the following code for menu items:

```
Sub mMenu11_Click ()
   Msgbox "Item1"
End Sub

Sub mMenu21_Click ()
   MsgBox "Item2"
End Sub
```

4. Add the following code to the Form_MouseDown event:

```
' Place following two lines of code on one, single line:
Sub Form_MouseDown (button As Integer, Shift As Integer, X As Single,
   Y As  Single)

   If button = 2 Then
      PopupMenu mMenu2
   Else
      PopupMenu mMenu1
   End If

End Sub
```

5. Run the application.

6. Click the client area of Form1. Then click the Item1 popup menu item.
   The correct message box will appear.

7. With Right mouse button, click the client area of Form1. The Item1
   message box will appear, indicating the click event was generated
   for the first menu item under top-level menu 1.

Additional reference words: buglist3.00 3.00 buglist3.00 hidden
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Num Lock Turned Off After Sending Keystrokes to DOS App.

Article ID: Q118818

The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for
   Windows, versions 2.0 and 3.0
 - Microsoft Visual Basic Programming System for Windows, version 1.0

SYMPTOMS
========

After you activate an MS-DOS application in a window and send keystrokes to it, the Num Lock key, which had been turned on, is turned off.

WORKAROUND
==========

You can use the KeyStat control to work around the problem. The following code fixes the example in the "MORE INFORMATION" section, below:

```
   ' Add a KeyStat control (KeyStat1 by default)
   ' Set the Style property to 1 - Num Lock
   ' Set the visible property to False
   Sub Command1_Click ()
      NumLockStatus = KeyStat1.Value
      Clipboard.SetText "DIR" + Chr(13)
      AppActivate "MS-DOS Prompt" ' Title of Windowed MS-DOS Session
      SendKeys "% ep"
      KeyStat1.Value = NumLockStatus
   End Sub
```

NOTE: The KeyStat control does not fix the problem if you specify True for the Wait parameter in the SendKeys statement.

STATUS
======

Microsoft has confirmed this to be a bug in the Standard and Professional Editions of Visual Basic versions 2.0 and 3.0 for Windows and in the Visual Basic Programming System version 1.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Step-by-Step Example to Reproduce the Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a command button to the form (Command1 by default), and place the
   following code in the click event:

```
   Sub Command1_Click ()
      Clipboard.SetText "DIR" + Chr(13)
      AppActivate "MS-DOS Prompt" ' Title of Windowed MS-DOS Session
      SendKeys "% ep"
   End Sub
```

3. Open an MS-DOS window. The MS-DOS window should be in the "Normal" state (use ALT + ENTER if necessary).

4. If the title of the MS-DOS Window is not "MS-DOS Prompt", then modify the AppActivate statement in the command-button click event to read that way.

5. Start the application.

6. Turn Num Lock on.

7. Press the command button. SendKeys sends the keystrokes "DIR" to the MS-DOS window. The Num Lock key gets turned off in the process.

Additional reference words: buglist1.00 buglist2.00 buglist3.00 1.00 2.00
3.00 NumLock
KBCategory: kbenv kbbuglist
KBSubCategory: EnvtRun

BUG: WDCONST.BAS File Described in ODK Docs Doesn't Exist
Article ID: Q118820

The information in this article applies to:

 - Professional Edition of Microsoft Visual Basic for Windows,
   version 3.0
 - Microsoft Office Developer's Kit, version 1.0

SYMPTOMS
========

The book "Programming Integrated Solutions" that comes with the Office Developer's Kit has a description of the file WDCONST.BAS in chapter 6, "Microsoft Word Object," in the "Constants" section, pages 173 and 174. It says that WDCONST.BAS includes definitions of constants that you can use in Visual Basic version 3.0 when calling Word for Windows version 6.0 OLE automation methods. However, when you look for this file on the compact disc, it is not there.

CAUSE
=====

The WDCONST.BAS file was never created and the documentation was not changed to reflect this.

WORKAROUND
==========

You can define your own constants as documented in the example code in the section of "Programming Integrated Solutions" mentioned above.

STATUS
======

Microsoft has confirmed this to be a bug in the Office Developer's Kit version 1.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 1.00 3.00 constant ODK docerr
KBCategory: kbref kbdocerr kbbuglist
KBSubcategory: RefsDoc

BUG: No Error Produced when Data Changed in DataControl

Article ID: Q119244

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0
- Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer

SYMPTOMS

This article describes a scenario where two or more Visual Basic applications using the Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer to edit and update the same record at the same time fail to produce error message 3197: "Data has changed; operation stopped."

STATUS

Microsoft has confirmed this to be a bug in the Standard and Professional Editions of Visual Basic for Windows, version 3.0, and the Jet 2.0/Visual Basic 3.0 Compatibility Layer. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. On a computer with the Visual Basic 3.0 Compatibility Layer installed, start a new project in Visual Basic. Form1 is created by default.

2. Add a text box (Text1), command button (Command1), and data control (Data1) to the form.

3. Set the following properties of each of the controls according to the table listed below:

Control	Property	Setting	Comment
Data1	DatabaseName	BIBLIO.MDB	Sample MDB in the VB directory.
Data1	RecordSource	Authors	The Authors table is in BIBLIO.MDB.
Text1	DataSource	Data1	Bind the text box to the data control.
Text1	DataField	Author	Put author's name in the text box.
Command1	Caption	Update	What the button does.

4. Place the following code in the Click() event for the command button:

```
      Sub Command1_Click()
         Data1.RecordSet.Update
      End Sub
```

5. Choose Make EXE File from the File menu, and compile the program as "DATAERR.EXE". Choose OK.

6. Exit Visual Basic, saving and naming the .MAK and .FRM files. In the Program Manager, choose Run from the File menu. Type "C:\VB\DATAERR.EXE" in the prompt and choose OK. Repeat this step so that you have two instances of the application running.

7. In the first instance, make a change to the author's name and choose Update (do not choose the data control).

8. In the second instance, make a different change to the author's name and choose Update. Notice that you did not get error message 3197 "Data has changed; operation stopped" as you should have.

WORKAROUND
==========

To work around this problem, add the following lines of code to your Form_Load Event().

```
   Data1.Refresh        'Make sure the DatabaseName and RecordSource
                        'properties are set before you do this.
   Data1.RecordSet.Edit
   Data1.RecordSet.Update
```

Additional reference words: 3.00
KBCategory: kbprg kbcode kbbuglist
KBSubcategory: APrgDataAcc

BUG: Error with Edit Method After Rollback of Previous Edit

Article ID: Q119733

The information in this article applies to:

 - Microsoft Visual Basic Programming System for Windows, version 3.0

SYMPTOMS
========

You may receive an "Update without AddNew or Edit" or "No current record" run-time error message when using the Edit method on a dynaset, if you have previously used Rollback to roll back a transaction that contained an Edit method without a corresponding Update method.

WORKAROUND
==========

To work around this problem, you can either ensure that you perform an Update before issuing the Rollback or use a "dummy" AddNew method after the Rollback.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

To reproduce this problem, perform the following steps:

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a command button (Command1) to the form.

3. Add the following code to the Command1_Click event:

```
     Sub Command1_Click()
       Dim db As database
       Dim ds As dynaset

       Set db = OpenDatabase("biblio.mdb")
       Set ds = db.CreateDynaset("select * from authors")
       ds.Filter = "au_id > 1"
       Set ds = ds.CreateDynaset()

       BeginTrans
       ds.Edit

       'ds.update    ' This is the irst workaround.

       Rollback
       'ds.addnew    ' This is the second workaround.

       ds.Edit       ' The error occurs here.

     End Sub
```

5. Press F5 to run the code.

An "Update without AddNew or Edit" error message will display on the second ds.Edit statement. (If you have the Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer installed, you may receive a "No current record" error at this point instead.)

To work around this problem, you can perform an Update immediately before the Rollback. This will have no permanent effect: the Rollback will undo the Update operation because the Update is inside the transaction. Another option is to perform an AddNew before the first Edit after the Rollback. (The AddNew will be voided by the Edit because the Edit occurs before an Update.)

Additional reference words: buglist3.00 3.00
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: Recompiling VB Project May Produce Larger .EXE File
Article ID: Q119734

The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for
   Windows, version 3.0

SYMPTOMS
========

When you compile a Visual Basic project (by choosing Make EXE File from the File menu) more than once without making any modifications to the project, the size of the subsequent .EXE file may be different. In other applications, successive compilation of an application's unaltered files results in executable files of the same size. However, in Visual Basic the .EXE file usually increases in size with each compilation.

CAUSE
=====

Each Visual Basic application receives one data segment of up to 64K (minus overhead) to store global variables and global constants. Space for the global string constant descriptors is allocated in this data segment. The actual text for global string constants is stored in a segment of up to 32K (minus overhead), allocated separately from dynamic memory.

A Visual Basic custom control (.VBX) is allocated space in the same 32K segment for any strings created with the Visual Basic API VBCreateHlstr. If the custom control does not deallocate this space, because it needs to reference these strings when it is unloaded or because it does not clean up properly, the data in the segment may exceed 32K. Visual Basic allocates another segment to hold the excess data any time the data exceeds 32K and links this new segment to an internal list of dynamic data segments.

While it is legal to allocate another segment to custom controls, the global string constant data is limited to one segment. Visual Basic gets confused because the global string constant data segment has another segment attached to it, and an .EXE file of a larger size is usually generated as a result. In some cases, the global string constant data segment is even lost, resulting in a smaller .EXE file.

WORKAROUND
==========

If you encounter the above situation, the only way to make consistent .EXE files so that any differences in size can be attributed to a problem like the one described in this article is to compile from the command line. In Program Manager, choose Run from the File menu and enter a Visual Basic command line like the one below in the Command Line text box:

```
     VB.EXE /Make SAMPLE.MAK SAMPLE.EXE
```

Because it is impossible to tell how much of the segment custom controls using VBCreateHlstr may need and because Visual Basic needs some of the space to manage the segment, you should also try to use as little space as possible for global string constants, thus attempting to avoid allocating another segment. To eliminate this possibility, you should try and reduce the space needed to store global string constant data to well under 32K. This space can be reduced by using fewer global string constants and by making the strings shorter.

STATUS
======

Microsoft has confirmed this to be a bug in Visual Basic version 3.0. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Executable files compiled from the command line for the same unmodified project are normally the same size, but they are not necessarily identical. Executable files produced by Visual Basic contain some uninitialized data, and when they are saved, they contain whatever happened to be at that location on the disk. You will see these differences within the .EXE files when comparing them with a binary-file comparison utility like DIFF.

For further information on what Visual Basic places in its .EXE files, please see the following article in the Microsoft Knowledge Base:

```
   ARTICLE-ID: Q112860
   TITLE     : General Memory Management in Visual Basic 3.0 for Windows
```

Additional reference words: buglist3.00 3.00 inconsistent
KBCategory: kbenv kbbuglist
KBSubcategory: EnvtDes

BUG: Multiline Text Box Incorrectly Displays Large Text
Article ID: Q120523

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0

SYMPTOMS
========

A MultiLine Text Box control with only a vertical scroll bar may
incorrectly display large strings that do not contain spaces or line feed
characters. The control will potentially display an incorrect string, no
string at all, or modify the visible font without affecting the control's
font property.

STATUS
======

Microsoft has confirmed this to be a problem with the Windows Text Box
(Edit) control. We are researching this problem and will post new
information here in the Microsoft Knowledge Base as it becomes available.

WORKAROUND
==========

To workaround this problem, set the ScrollBars property of the Text box to:

'1 - Horizontal' or '3 - Both'

MORE INFORMATION
================

Steps to Reproduce Behavior

The following test application will reproduce the problems described above.
Variations in FontName and FontSize will produce different results.

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a Text Box (Text1) and three Command Buttons (Command1,
   Command2, and Command3) to Form1.

3. Set the Multiline property of Text1 to True and set the ScrollBars
   property to 2 - Vertical.

4. Add the following code to the load event of Form1:

```
   Sub Form_Load ()
      Text1.Left = 100
      Text1.Top = 100
      Text1.Width = 4000
      Text1.Height = 1600
      Command1.Left = 4500
      Command1.Top = 100
      Command2.Left = 4500
      Command2.Top = Command1.Top + Command1.Height + 25
      Command3.Left = 4500
      Command3.Top = Command2.Top + Command2.Height + 25
   End Sub
```

5. Add the following code to the click event of Command1:

```
   Sub Command1_Click ()
      Dim A As String
      Dim B As String

      'A single '1' will appear in the Text
      'box and the Font will not be bold although
      'the property is set to True

      Text1.Text = ""
      Text1.FontName = "MS Sans Serif"
      Text1.FontBold = True

      A$ = String(4096, "1")
      B$ = String(4096, "2")

      A$ = A$ & " " & B$
      Text1.Text = A$
   End Sub
```

6. Add the following code to the click event of Command2:

```
   Sub Command2_Click ()
      Dim A As String
      Dim B As String

      'No text will appear in the Text1

      Text1.Text = ""
      Text1.FontName = "MS Sans Serif"
      Text1.FontBold = True

      A$ = String(4096, "1")
      B$ = String(4096, "2")

      A$ = A$ & B$
      Text1.Text = A$
   End Sub
```

7. Add the following code to the click event of Command3:

```
   Sub Command3_Click ()
      Dim A As String
      Dim B As String

      'A blank line will appear followed by the 'Y's

      Text1.Text = ""
      Text1.FontName = "MS Sans Serif"
      Text1.FontBold = True

      A$ = String(4096, "X")
      B$ = String(4096, "Y")

      A$ = A$ & B$
      Text1.Text = A$
   End Sub
```

8. Save the project and run. Click each of the command buttons to see
   the following results:

    - Command1: A single 1 will appear in the text box and the font will
                not be bold.

    - Command2: No text will appear in the text box.

    - Command3: A single blank row or row of Xs followed by Ys will be
                visible in the text box.

Additional reference words: 2.00 3.00 buglist2.00 buglist3.00
KBCategory: kbui kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: Wrong Menu with Maximized MDI Child and No Control Box
Article ID: Q121096

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, versions 2.0 and 3.0

SYMPTOMS
========

If you have a maximized MDI child form that doesn't contain a ControlBox
(ControlBox = False), references to the MDI parent menus will control the
wrong menu.

WORKAROUND
==========

The only way to remedy this problem is to set the MDI child form's
ControlBox property to True.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic versions
2.0 and 3.0 for Windows. We are researching this problem and will post new
information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic and create a new project.

2. Add an MDI form to the project

3. Add a menu to the MDI form using the following as a guide:

```
   Caption            Name
   --------------------------
   Menu1              mMenu1
   ....Menu1Item1     mMenu11
   Menu2              mMenu2
   ....Menu2Item1     mMenu21
```

4. Add the following code for menu items:

```
   Sub mMenu11_Click ()
      Form1.Show
   End Sub
```

5. Set Form1.MDIChild = True and Form1.ControlBox = False

6. Add the following code to Form1:

```
   Sub Form_Click()
      MDIForm1!mMenu1.enabled = not MDIForm1!mMenu1.enabled
   End Sub
```

7. From the Options Menu, choose Project Options. Change the startup form to MDIForm1

8. Run the program. Choose Menu1Item1 from Menu1. The MDI child should appear. Click the MDI child form. This disables Menu1. Maximize the MDI child, and click it again. Now Menu2 is disabled.

Additional reference words: 2.00 3.00 buglist2.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: GP Fault When KEYSTAT.VBX Used in Two or More Apps
Article ID: Q121681

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  version 3.0

SYMPTOMS
========

When the KEYSTAT.VBX control is used in more than one application, and the parent window is minimized, the application causes a general protection (GP) fault.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft product listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

WORKAROUND
==========

To work around this problem, use the Windows API GetKeyboardState function directly as in the following example:

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following code to the (general) (declarations) section

```
   Declare Sub GetKeyboardState Lib "User" (ByVal lpKeyState As String)
   Const VK_SHIFT = &H10
   Const VK_CAPITAL = &H14
   Const VK_NUMLOCK = &H90
```

3. Place a command button (Command1) on Form1, and add the following code to the Command1 Click event:

```
   Sub Command1_Click ()
      Dim keys As String
      keys = Space$(256)
      GetKeyboardState keys
      If &H1 = (Asc(Mid(keys, VK_CAPITAL + 1, 1)) And &H1) Then
         Debug.Print "Caps lock"
      Else
         Debug.Print "non caps"
      End If
      If &H1 = (Asc(Mid(keys, VK_NUMLOCK + 1, 1)) And &H1) Then
         Debug.Print "Num lock"
      Else
         Debug.Print "non lock"
      End If
   End Sub
```

4. Run the application.

Additional reference words: 3.00 buglist3.00 gpf
KBCategory: kbprg kbbuglist kbcode
KBSubcategory: PrgCtrlsCus

BUG: GPF in VBDB300.DLL When Use ODBC to Connect to Oracle DB

Article ID: Q124503

The information in this article applies to:

 - Professional Edition of Microsoft Visual Basic for Windows,
   version 3.0

SYMPTOMS
========

When trying to use ODBC with Visual Basic for Windows to get table information from an Oracle database, you may receive the following error message:

```
   An error has occurred in your application...
```

When you choose the Close button, the following general protection (GP) fault error message appears:

```
   VB caused a GPF in module VBDB300.DLL at 000A:1B0F
```

If you choose Ignore, your application may continue to run as expected.

CAUSE
=====

It appears that this GP fault is caused within the file VBDB300.DLL by an incompatibility with specific network configurations. However, neither Microsoft nor Oracle have been able to duplicate this problem.

The problem has been reported to occur with both Oracle 6 and Oracle 7 on a variety of network software with a variety of network protocols. There is no one specific network or protocol that will reproduce the problem.

WORKAROUND
==========

The following workaround has been found to be successful if in addition to Visual Basic, you also have both Microsoft Access version 2.0 and the Microsoft Access 2.0/Visual Basic 3.0 Compatibility Layer installed:

In the VB.INI or <appname>.INI file (see page 148 of "Professional Features book 2"), add the following to the [ODBC] section (create one if needed):

```
   [ODBC]
   AttachableObjects='TABLE'
```

For more information about this .INI file entry, please see the Help menu in Microsoft Access version 2.0. Search for "INI Files," and select the "Customizing MSACC20.INI Setting" topic.

For additional information about the Compatibility Layer and how to get it, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: Q113951
TITLE   : How to Obtain & Distribute the Compatibility Layer

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem

If you have a network configuration on which the problem does occur, the following piece of code will demonstrate the problem:

```
   Sub Command_Click()
      Dim db as Database
      Dim ds as Dynaset
      Set db = OpenDatabase("", 0, 0, "ODBC;")
         ' You will be prompted for the ODBC server information.
      Set ds = db.CreateDynaset("<tablename>")
   db.TableDefs.Refresh
   End Sub
```

The error occurs on the last line of code where the Refresh is trying to load the table information from the Oracle database into memory.

Additional reference words: 3.00 gpfault buglist3.00 gpf
KBCategory: kbprg kbbuglist kberrmsg
KBSubcategory: APrgDataODBC

BUG: (CDK) Cannot Use an Underscore in a Custom Event Name

Article ID: Q126221

The information in this article applies to:

- Microsoft Visual Basic Control Development Kit (CDK) for
  Microsoft Visual Basic Programming system for Windows, version 1.0
- Microsoft Professional Toolkit for Microsoft Visual Basic,
  version 1.0
- Professional Edition of Microsoft Visual Basic for Windows,
  versions 2.0 and 3.0

SYMPTOMS
========

A custom Event containing an underscore character (_) as part of the Event's name will not fire, and it produces duplicate code in the Microsoft Visual Basic Development Environment.

Also, if code is placed in the custom event that has an underscore in its name, a duplicate of the event code will appear in the general section of the IDE.

CAUSE
=====

If an underscore is present in an event's name, the return value from a call to VBFireEvent will be Zero, but the custom event will not be triggered in Visual Basic. This is because the underscore prevents Visual Basic from finding the name of the event in the event list.

The reason for the duplicate code is that Visual Basic is unable to match the code with the event for that control due to the underscore in the event name.

WORKAROUND
==========

Do not use the underscore character as part of the name of a custom event.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Create a VBX control that has an underscore character in its event name.

2. Load the VBX into a Visual Basic project.

3. Bring up a code window the form containing the control in question.

4. Choose [general] from the [Object:] drop down list.

5. Select the [Proc:] drop down list.

6. As you scroll the listbox, you will see duplicates of the functions and' code you thought were associated with the control event.

Additional reference words: 1.00 2.00 3.00 buglist1.00 buglist2.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

BUG: Menu Lost if Caption Changed on Menu with Only WindowList
Article ID: Q126675

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

When a top-level menu has its WindowList property set to True and the top-level menu does not contain any sub-menus, after changing the menu's caption the top-level menu will no longer display the list of MDI child windows and you will not receive any click events.

RESOLUTION
==========

In order to work around the problem, make sure the top-level menu item that has the WindowList property set to True has at least 1 sub-menu item.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION
================

In the example below, a menu on the MDI form contains nothing but a WindowList. A command button is placed on the MDI child form with code behind it to change the caption of the menu containing the WindowList. As long as you do not click on the command button, the Window menu will display a list the MDI child windows. Once you click on the command button, the Window menu caption is changed to "Test" and it no longer keeps track of the MDI child windows and it no longer receives click events.

NOTE: Issuing the command "Print MDIForm1.mnuwindow.Windowlist" (without the quotes) from the immediate window will return -1 (True). However, the Window list is not displayed.

Steps To Recreate The Problem
----

1. Start a new project in Visual Basic (ALT, F, N). Form1 is created
   by default.

2. From the Property Window, set MDIChild equal to True for Form1.

3. Add a Command button (Command1) to Form1 and add the following

code to the Click event.

```
Sub Command1_Click ()
   MDIForm1.mnuWindow.Caption = "Test"
End Sub
```

4. From the File menu, select New MDI Form. MDIForm1 is created by default.

5. From the Window menu, choose Menu Design.

6. Create a menu for the MDI Form with the following properties:

Caption	Name	Indented	WindowList
File	mnuFile	No	Unchecked
New	mnuNew	Once	Unchecked
Window	mnuWindow	No	Checked

7. Choose Done to exit the Menu Design Window. You now have two top-level menus.

8. Add the following code to the Click event of mnuNew menu on MDIForm1.

```
Sub mnuNew_Click ()
  Dim f As New Form1
  f.Show
End Sub
```

9. Press F5 to run the program.

10. The Window menu will keep track of all the MDIChild windows as long as you do not click Command1 contained in the MDIChild. Once you click on the command button, the Window menu's caption has changed to "Test" and it will no longer list any of the MDIChild windows and you will not receive any click events.

Additional reference words: 3.00 buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsStd

BUG: RegisterDatabase Fails After ODBC Version 2.x Installed
Article ID: Q126940

```
---------------------------------------------------------------------
The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  version 3.0
---------------------------------------------------------------------
```

SYMPTOMS
========

The RegisterDatabase function fails and returns error 3146 "ODBC call failed" after ODBC version 2.x has been installed.

CAUSE
=====

The ODBC API function SQLConfigDataSource() was not correctly implemented in ODBC version 1.0. The RegisterDatabase function in Visual Basic version 3.0 was designed to use the ODBC version 1.0 implementation of SQLConfigDataSource().

SQLConfigDataSource() is now implemented correctly in ODBC version 2.x. If ODBC version 2.x has been installed you will find that an application that uses RegisterDatabase will fail with the error 3146 "ODBC call failed."

Visual Basic exhibits this problem when using RegisterDatabase because it was based on the original implementation of SQLConfigDataSource(). The underlying function that RegisterDatabase ultimately calls has been changed and Visual Basic's RegisterDatabase function will now generate the error if the ODBC version 2.x DLLs are installed.

WORKAROUND
==========

You can code directly to the ODBC API and register your DSN with SQLConfigDataSource(). Below is a code sample that will replace the functionality of Visual Basic's RegisterDatabase function. It first tries to use RegisterDatabase, if that fails, it uses the ODBC API function SQLConfigDataSource. You can call this function instead of calling RegisterDatabase.

Step-by-Step Example
----

1. In Visual Basic, create a new module with the following declarations:

```
   Option Explicit
   Const ODBC_ADD_DSN = 1         ' Add a new data source.
   Const ODBC_CONFIG_DSN = 2      ' Configure (edit) existing data source.
   Const ODBC_REMOVE_DSN = 3      ' Remove existing data source.

   ' Enter the following three lines as one, single line:
   Declare Function SQLConfigDataSource Lib "odbcinst.dll"
      (ByVal hwnd as Integer, ByVal fRefresh as Integer,
      ByVal szDriver as String, ByVal szAttributes as String) As Integer
```

2. Create the following procedure.

```
   ' Enter the following two lines as one, single line of code:
   Sub RegisterODBCDatabase (dsn As String, driver As String,
      silent As Integer, attributes As String)

      Dim ret As Integer
      On Error GoTo errorhandler

      RegisterDatabase dsn, driver, silent, attributes
   Exit Sub

errorhandler:
   If Err = 3146 Then    ' ODBC Call Failed.
      Dim temp As String
      Dim spot As Integer

      While InStr(attributes, Chr(13))     ' Replace Carriage returns
         spot = InStr(attributes, Chr(13)) ' with nulls.
         Mid(attributes, spot, 1) = Chr(0)
      Wend
      attributes = attributes & Chr(0) & Chr(0) ' End of attribute section.
      temp = "DSN=" & dsn & Chr(0) & attributes

      ret = SQLConfigDataSource(0, ODBC_ADD_DSN, driver, temp)

      ' ret is equal to 1 on success and 0 if there is an error.
      If ret <> 1 Then
         MsgBox "SQLConfigDataSource call failed"
      Else
         MsgBox " SQLConfigDataSource call succeeded!"
      End If
   End If
   Exit Sub
   End Sub
```

3. To test this procedure, press the F8 key to single step. Then activate the Debug window from the Window menu. Type the following, and press the ENTER key:

```
   RegisterODBCDatabase "SqlServer", "TestSQL", "MyServer", "Pubs"
```

As a result, you will receive a message stating the new datasource was added successfully.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

1. Start a new project in Visual Basic. Form1 is created by defalt.

2. Add a Command button (Command1) to Form1 and add the following code to
   the Command1 button's Click event:

```
   Sub Command1_Click ()
      Dim Att As String, MyDb As Database
      Att = "Description = SQL Server on server Clinton" & Chr$(13)
      Att = Att & "OemToAnsi=No" & Chr$(13)    ' Build keywords string.
      Att = Att & "Network=DBNMP3" & Chr$(13)
      Att = Att & "Address=\\CLINTON\PIPE\SQL\QUERY" & Chr$(13)
      Att = Att & "Database=Pubs"
      ' Update ODBC.INI.
      RegisterDatabase "Clinton", "SQL Server", True, Att
   End Sub
```

3. Press the F5 key to run the program. If ODBC version 2.x has been
   installed, when you click the command button, you will receive this
   error: "ODBC Call Failed."

REFERENCES

"Microsoft ODBC 2.0 Programmer's Guide and SDK Guide" Chapter 24.

Additional reference words: 3.00
KBCategory: kbprg kbbuglist kbcode
KBSubcategory: APrgDataODBC

BUG: GP Fault in VBRUN300.DLL at 005D:2332

Article ID: Q126991

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 3.0

SYMPTOMS
========

You may encounter a general protection (GP) fault in VBRUN300.DLL at 005D:2332 when you try to run a Visual Basic executable (.EXE) file that has a Data control. The GP fault occurs after the form's Load event but before the Activate event; it is during this period of time that the Data control is automatically refreshed.

WORKAROUND
==========

To work around the problem, refreshing the data control explicitly in the Load event of the form using code like this:

```
  Sub Form_Load()
     Data1.Refresh
  End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 3.0 for Windows. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 3.00 buglist3.00 gpfault gpf
KBCategory: kbprg kbbuglist
KBSubcategory: APrgDataAcc

BUG: GP Faults from Using IIF with Temporary Strings
Article ID: Q127069

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0

SYMPTOMS
========

Using the IIF statement may cause a general protection (GP) fault in a range of addresses from 0023:6BB0 to 0023:7DE0 within VBRUN300.DLL or from 004A:68D0 or 004a:7B00 within VB.EXE. This is a subtle bug that may not be encountered and will not occur on a line using the IIF statement.

CAUSE
=====

The IIF statement does not release string handles for temporary strings used in its arguments. Under certain conditions, this can result in a GP fault occurring at a seemingly random location later in your program. The code below demonstrates the condition where IIF will not release its string handles.

```
   dim x$
   x$ = "World"
   y$ = "Hello World"
   Label1.Caption = IIf(3 > 1000, "Hello " & x$, Mid$( y$, 1, 5 ) )
```

The second and third arguments of this statement result in temporary strings being created. This does not, however, cause the GPF. One of two other events must occur for that to happen:

- If the user allocates a large string and there is not enough contiguous space for it, Visual Basic will attempt to compact the heap, creating a condition in which the GP fault can occur.

  -or-

- Upon exit of the procedure the IIF was located in, the heap will be compacted.

Once either condition occurs, the GP fault will not happen until the next temp string is allocated, which just happens to use the string handle used by the IIF statement. Then the GP fault will occur within one of the two address ranges given above.

WORKAROUND
==========

Avoid the use of functions that return Strings withing the IIF statement. For example, to avoid the bug, modify the code listed in the Cause section of this article to this:

```
   dim y$, z$
   y$ = "Hello World"
   z$ = "Small World"
   Label1.Caption = IIf(3 > 1000, y$, z$ )
```

Here's another alternative:

```
   if 3 > 1000 then
      label1.caption = y$
   else
      label1.caption = z$
   end if
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 3.00 GPF buglist3.00
KBCategory: kbprg kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: VB Debug.Print in MouseMove Event Causes MouseMove Event
Article ID: Q72679

The information in this article applies to:

 - Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Debug.Print used within the MouseMove event procedure of a form or control causes a MouseMove event. If the mouse cursor is located within the form or control, an endless stream of output to the Immediate Window will occur. This behavior occurs for a program run in the Visual Basic development environment. An .EXE program does not utilize the Immediate Window and the Debug object so this behavior does not apply to a .EXE program. The problem does not occur if a Print method is issued to any other form or control in the program.

STATUS
======

This is not a problem with Visual Basic, but rather the nature of the Microsoft Windows operating environment. This problem does not occur in Visual Basic version 2.0 or 3.0.

MORE INFORMATION
================

If Debug.Print is used within the MouseMove event procedure of a form or control, an endless stream of output is sent to the Immediate Window. This occurs whenever the mouse cursor is within the form or control. This behavior occurs because the Debug.Print statement causes the focus to change briefly to the Immediate Window. When the focus returns to the form or control, Windows generates a MouseMove event that is processed by Visual Basic. There is no way for Visual Basic to suppress MouseMove events that are generated by Windows. The easiest way to overcome this behavior is to send debug output to another form or control.

To duplicate this behavior, create a picture control (Picture1) within the default form (Form1). Add the following code segment to the MouseMove event procedure of Picture1:

```
     Sub Picture1_MouseMove (Button As Integer, Shift As Integer,
                             X As Single, Y As Single)
     ' You must write the above Sub statement on just one line.
        Static i%
        i% = i% + 1
        Debug.Print i%
     End Sub
```

If you want output to be sent only when the mouse is moved, then all Debug.Print statements within the MouseMove event procedure should be changed to Print methods to other forms or controls. Below is a description of how to modify the example above such that output is produced only when the mouse is moved.

Add another form (Form2) to the project by selecting New Form from the File menu (ALT F+F). Change the Debug.Print statement in the MouseMove event procedure for Picture1 to Form2.Print. Below is a copy of the above sample modified to send output to another form.

```
     Sub Picture1_MouseMove (Button As Integer, Shift As Integer,
                             X As Single, Y As Single)
     ' You must write the above Sub statement on just one line.
         Static i%
         i% = i% + 1
         Form2.Print i%
     End Sub
```

In the example above, all output that scrolls off the form will be lost. A more sophisticated routine will be required to keep track of all output to the form. Such a routine is beyond the scope of this article.

Additional reference words: fixlist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist
KBSubcategory: EnvtDes PrgCtrlsStd

FIX: Overflow in VB Drawing Circle Segment w/ Radius of Zero

Article ID: Q73280

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When using the Circle statement to draw a segment of a circle with a radius of 0, an "Overflow" error incorrectly occurs.

STATUS

Microsoft has confirmed this to be a problem in Microsoft Visual Basic programming system for Windows, versions 1.0 and 2.0. This problem was corrected in version 3.0.

MORE INFORMATION

The following statement demonstrates the problem:

```
   Circle (0,0), 0,, 4, 5
```

When you run the above statement, an "Overflow" error incorrectly occurs.

In contrast, using the Circle statement to draw an entire circle of radius 0 works correctly without an error (correctly drawing nothing); for example:

```
   Circle (0,0), 0
```

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtRun

FIX: UAE When Place More than 64K in VB List Box or Combo Box
Article ID: Q73374

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows, versions 3.0 and 3.1

SYMPTOMS
========

Attempting to add more than 64K of data into a Visual Basic list box or combo box will result in a Windows Unrecoverable Application Error (UAE), when running under Windows version 3.0.

RESOLUTION
==========

This problem does not occur when running under Windows version 3.1. However, attempting to add more than 64K of data into a Visual Basic list box or combo box will result in an "out of memory" error message, when running under Windows, version 3.1.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic programming system for Windows, versions 1.0 and 2.0. We are researching this problem and will post new information here as it becomes available.

MORE INFORMATION
================

Each item of a list box or combo box can contain a string up to 1K in length; however, if the total of all items exceeds 64K, a UAE will be generated. The .List() property for list boxes and combo boxes is given its own segment up to 64K in size. If an attempt to exceed this limit is made, an "Out of memory" or "Out of string space" error message should result, but instead a UAE occurs and the program terminates.

Steps to Reproduce Problem
----

1. Create a New Project.

2. Draw a list box on Form1.

3. Add the following code to Form1's Click() event procedure:

```
   Sub Form_Click()
      Do
```

```
         List1.Additem String$(1024, "X")
         I = I + 1
         Debug.Print I
      Loop
   End Sub
```

When the UAE occurs, note that the value of the variable "I" displayed in the Immediate window will be 63. The UAE occurred when adding the 64th item, which caused the total size of the data in the list box to exceed 64K. The actual limit is slightly under 64K due to a small amount of overhead to manage the .List() property because it is a property array.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtRun

FIX: Pull-Down on Drive Box Disabled When Change Width of Box

Article ID: Q73809

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you change the Width property of a drive list box at run time, the pull-down list of drives no longer work.

WORKAROUND
==========

Add the following code to the form's click event procedure to work around the problem:

```
Sub Form_Click ()
   Drive1.Width = Drive1.Width * 2
   Drive1.Refresh                         '* fixes the problem
End Sub
```

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic programming system for Windows, versions 1.0 and 2.0. This bug was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic. Form1 is created by default.

2. Add a drive list box to Form1.

3. In the Click event of Form1, add the following code:

```
   Sub Form_Click ()
      Drive1.Width = Drive1.Width * 2
   End Sub
```

4. Run the application (press F5).

5. Click the down arrow on the drive box to display the list.

6. Choose a drive; everything works as it should.

7. Click Form1; the width of the drive box changes.

8. Click the down arrow on the drive box.

Note that the list fails to display.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: UAE/GPF Changing MS-DOS Win Display If VB at Breakpoint

Article ID: Q74193

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When using some Windows display drivers, the following steps may cause Microsoft Visual Basic to abort with an Unrecoverable Application Error (UAE) in Windows version 3.0 or a general protection (GP) fault in Windows version 3.1:

1. Start Microsoft Visual Basic.

2. Add a line of code to the Form_Click event procedure, such as "X = 5".

3. Set a break point on the line added in step 2.

4. Start a simultaneous MS-DOS session in Windows.

5. Run the Visual Basic program (F5); click the form to stop at the break point.

6. Activate (double-click) the MS-DOS window.

7. Press ALT+ENTER to change the MS-DOS window to a full screen window.

Pressing ALT+ENTER to change the MS-DOS window to a full screen MS-DOS session may result in a UAE or GP fault.

STATUS
======

This behavior is a result of problems with certain Windows display drivers, and not a problem with Visual Basic. This problem does not occur in Visual Basic version 2.0 or 3.0 for Windows.

Additional reference words: fixlist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Overflow Error If Print Long String to Form or Printer

Article ID: Q74517

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0
---------------------------------------------------------------------
```

SYMPTOMS
========

An "Overflow" error message may occur when you print a long string in Microsoft Visual Basic for Windows.

When a character is printed using the Print method, the CurrentX and CurrentY coordinates are also updated for the object being printed to. If the string being printed is long enough to cause the value of the CurrentX property to exceed 32,767 twips, an "Overflow" error will occur. This behavior is by design.

"Overflow" can be caused by printing a single long string or by repeatedly printing shorter strings that are appended onto the end of the last string -- using the Visual Basic semicolon (;) operator.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic versions 1.0 and 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic or choose New Project from the File menu.

2. Place a label control on Form1.

3. Add the following code to the Form_Click event:

```
        Sub Form_Click()
            For index% = 1 to 1000
                Print "A";
                Label1.Caption = Str$(CurrentX)
            Next
        End Sub
```

4. From the Run menu, choose Start.

5. Click Form1.

An "Overflow" error will occur. You can examine the label caption to
see that the value of Form1.CurrentX plus the TextWidth of "A"
exceeded 32767 at the time of the error.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00
3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: Control Overlaid by 2nd Control Won't Refresh If Moved

Article ID: Q74519

The information in this article applies to:

 - Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Visual Basic version 1.0 for Windows does not support overlapping controls. Having overlapping controls can result in portions of a control not refreshing correctly. If controls are moved over each other, then one or both of the controls may not correctly refresh even when the controls are moved apart. This is known to happen when controls are resized at run time using the Move method or by changing the Height and Width properties as a result of a Form_Resize event. Because controls must be resized one at a time, it is possible that one control will briefly overlap another control during the resize process at run time. The control that was briefly overlapped may not refresh properly. An example of this behavior is given in the More Information section below.

WORKAROUND
==========

This behavior can be improved by performing the Refresh method (CtrlName.Refresh) on every overlapping control at run time, after an overlapped control has been moved or after a form that contains overlapping controls has been resized.

STATUS
======

This is not a problem with Visual Basic. It is the nature of overlapping controls in Visual Basic version 1.0. This behavior occurs at run time in the Visual Basic development environment or as an .EXE program.

This problem does not occur in Visual Basic version 2.0 or 3.0 for Windows where overlapping controls are supported.

MORE INFORMATION
================

For more information about Visual Basic and overlapping controls, query in this knowledge base on the following words:

```
   overlapping and controls and Visual and Basic
```

Steps to Reproduce Problem
----

1. From the File menu, choose New Project (ALT, F, P).

2. Add a picture control (Picture1) to the default form (Form1).

3. Add a command button (Command1) to Form1.

4. Add a vertical scroll bar (VScroll1) to Form1.

5. Using the mouse, double-click Form1 to bring up the code
   window.

6. Within the Resize event procedure of Form1, add the following code:

```
     Sub Form_Resize ()
         Picture1.Move 0, 0, ScaleWidth - VScroll1.Width, _
                       ScaleHeight - Command1.Height
         VScroll1.Move ScaleWidth - VScroll1.Width, 0, _
                       VScroll1.Width, ScaleHeight - Command1.Height
         Command1.Move 0, ScaleHeight - Command1.Height, _
                       ScaleWidth, Command1.Height
     End Sub
```

Note: The underscores (_) in the above code example indicate that
the line should be concatenated with the next line in the Visual Basic
environment (VB.EXE).

7. Run the program.

8. Using the mouse, resize the form by extending the bottom or right
   sides. When the bottom edge of the form is extended, the command
   button (Command1) will not refresh. When the right edge of Form1
   is extended, the scroll bar will not refresh. The refresh problems
   are caused because Picture1 is expanded and temporarily overlaps
   the control. When the control (VScroll1 or Command1) is moved out
   of the way, it is not refreshed.

To work around this behavior, use the Refresh method for Picture1,
VScroll1, and Command1 after the controls have be moved. Add the
following statements to Sub Form_Resize (after the Command1.Move
statement) above to overcome the behavior:

```
   Picture1.Refresh
   VScroll1.Refresh
   Command1.Refresh
```

Additional reference words: fixlist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Text Not Highlighted When Copy Immediate Win to Clipboard
Article ID: Q75762

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When you copy text from the Immediate window to the Windows system clipboard, the selected text is not highlighted. Also, the cursor is not visible. However, the copy operation works as it should.

STATUS
======

Microsoft has confirmed this to a bug in the products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce the problem:

1. Run the Windows Clipboard utility usually found in the Main group in Microsoft Windows Program Manager.

2. Start Visual Basic and press the F5 key.

3. Press CTRL+BREAK to bring up the Immediate window.

4. Press F5 to continue.

5. Click the Immediate window to give it the focus.

6. Press CTRL+HOME to move to the beginning of the text in the Immediate window.

7. Press SHIFT+CTRL+END to select all text in the Immediate window. Note that you cannot select text with the Mouse at this point.

8. Press CTRL+INS to copy the selected text in the Immediate window to the Windows clipboard.

The text goes onto the Windows clipboard as it should, but the text in the Immediate window is not highlighted as it should be, and the cursor is not visible.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist

KBSubcategory: APrgOther

FIX: Bad Text in Long Right-Aligned Labels in Windows ver 3.0

Article ID: Q76515

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows version 3.0

SYMPTOMS
========

When you use Visual Basic with Windows version 3.0, the caption of a right-aligned label that is set to a length exceeding 255 characters displays unusual (incorrect) characters. A left-aligned or centered caption displays correctly, and all captions display correctly when using Visual Basic with Windows version 3.1.

STATUS
======

Microsoft has confirmed this to be a problem in Windows version 3.0. This problem was corrected in Windows version 3.1.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. In the Visual Basic environment (VB.EXE), place a label on a blank form.

2. Add the following code to the form's Form_click event procedure:

```
   Label1.alignment = 1  'right justified
   Label1.caption = string$ (277, "k")
   Label1.refresh
```

3. From the Run menu, choose Start or press the F5 key.

4. Click anywhere inside the form except on the label to see unexpected characters appear in the rightmost portion of the caption.

Additional reference words: 1.00 2.00 3.00 garbage corrupted fixlist3.10
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: APrgOther

FIX: Undocumented Separator Property of a VB Menu Item

Article ID: Q76550

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

You may encounter an undocumented property for menu items that allows you to toggle between the caption and a separator bar. A separator bar in a menu is a horizontal line that separates menu item groups.

CAUSE

Microsoft did not intend to leave the Separator property in the Visual Basic product.

RESOLUTION

The Separator property is not documented in Visual Basic's manuals or online Help. Microsoft recommends that you not use the Separator property in your Visual Basic applications. The Separator property no longer exists in Visual Basic version 3.0.

STATUS

Microsoft has confirmed this to be a bug in the products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows by removing the Separator property.

MORE INFORMATION

```
Separator Property (Applies to Menu Items)
-------------------------------------------
Description: Denotes if a menu item is a separator bar. Prevents
             or allows the value of the Caption property of the
             menu item to be displayed versus a separator bar.

Usage:       [form.][menuitem.]Separator[= boolean%]

Remarks:     The Separator property settings are as follows:

             Setting        Description
             ------------------------------------------------------
             True (-1)      Disables the display of the value of
                            the Caption property as the menu item.
                            Instead, a separator bar is displayed.
```

```
False (0)     (Default) for all other menu items.
              The menu item will display the value
              of the Caption property for that item.
```

When a menu item is created in the menu design window, the value of its separator property defaults to 0 unless the caption of that menu item is set to a dash (-), in which case, the Separator property defaults to -1 and a separator bar is displayed for that item instead of the value of the Caption property.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Can't Have Menu with No Caption Bar/Buttons/Control Box

Article ID: Q76553

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

You can't add a menu that has no caption, no maximize/minimize buttons, and no control-menu box to a form.

CAUSE

This feature is not supported in Visual Basic in Windows version 3.0 or 3.1 because of a bug in the Microsoft Windows menu driver that prevents Windows from painting menus correctly.

STATUS

Microsoft has confirmed this to be a bug in Microsoft Windows versions 3.0 and 3.1. This problem does not occur in Visual Basic version 3.0 in Microsoft Windows version 3.1.

MORE INFORMATION

If you place a menu on a form with no caption bar or associated buttons, the result is a menu bar that does not refresh correctly.

Steps to Reproduce Problem

1. Start 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. Delete the contents of the Caption property.

3. Set the ControlBox, MaxButton, and MinButton properties to False.

4. Using the Menu Design window, create a single menu item. Set the Caption property to Test and the control name property to Test.

5. Press the F5 key to run the application.

Note how the menu bar does not repaint correctly. This causes the image immediately behind the form to be visible through the menu bar.

If you place any other form over the menu bar and then remove it, the portion that was covering the menu bar area remains.

This problem occurs because the Microsoft Windows menu driver does not paint the menus correctly.

For this reason, this particular form configuration is not supported by Visual Basic at this time even though you are able to create the configuration in the editing environment.

For more information about a related problem with the menu bar and the Fixed Double border style, query on the following words in the Microsoft Knowledge Base:

   Visual and Basic and menu and fixed and double and border

Additional reference words: buglist2.00 fixlist3.00 1.00 2.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: ControlBox Property False Disables Focus w/ Keys in Menus

Article ID: Q76556

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When the ControlBox property on a form is set to False, (disabling the Control Box), the ability to change focus within menus using the keyboard (such as by using the ARROW keys) is lost. This is because of a limitation of Windows; it is not a problem with Visual Basic.

STATUS
======

Microsoft has confirmed this to be a problem with Windows version 3.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

This problem only applies to changing focus between menu items. The ARROW keys work correctly to change focus with other controls (for example, two command buttons), even with the ControlBox disabled.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic.

2. From the Window menu, choose Menu Design Window.

3. Enter Test1 and Test2 as the caption and CtlName of two separate top level menu items. Choose the Done button to close the Menu Design window.

4. From the Properties box, select ControlBox.

5. From the Settings box, set the ControlBox property to False. (This removes the ControlBox from the form at run time.)

6. Press F5 to run the application.

Notice that the mouse can be used to select either the Test1 or Test2 menu, but pressing the ALT key followed by the LEFT or RIGHT ARROW keys will not allow you to move between the menus. You will only be able to select the Test1 menu by pressing the ALT key.

Setting the ControlBox property to True will re-enable the LEFT/RIGHT ARROW keys to select menu items.

Additional reference words: fixlist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Printing with HPPCL5A.DRV to HP LaserJet III Cuts Line

Article ID: Q78079

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

Choosing Print from the Visual Basic File menu to print source code truncates one line of code per page of output when printing to a Hewlett-Packard (HP) LaserJet series III printer using the HPPCL5A.DRV printer driver.

CAUSE
=====

This is a problem with the Hewlett-Packard LaserJet series III printer driver version 3.42 for Windows.

STATUS
======

Microsoft has confirmed this to be a problem with the HPPCL5A printer driver version 3.42. This problem was corrected by the HP III driver version 30.3.85 included with Microsoft Word for Windows version 2.0.

Additional reference words: 1.00 2.00 HP laser jet truncate lose
KBCategory: kb3rdparty kbhw kbenv kbprg kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Right Mouse Button Causes Remote Control Menus

Article ID: Q78773

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows version 3.0

SYMPTOMS
========

In Windows version 3.0, the mouse behaves unexpectedly in both Visual Basic version 1.0 and 2.0 under the following conditions:

- A Visual Basic program is run from the environment (VB.EXE) or from an executable using the run-time module.
- The program has a form that contains menus.
- While holding a menu open with the left button, if you click the right mouse button, the mouse selection appears to be inactivated.
- Moving up or down the menu while holding the left button down, causes no selection until you get several inches below the pop-up (or pull-down) menu. At that point, the mouse causes selection again from the remote position.

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article when used with Microsoft Windows version 3.0. This bug was corrected in Microsoft Windows version 3.1.

MORE INFORMATION
================

This problem occurs when running a Visual Basic program that has menus. It requires a mouse with two buttons and has been reported with both the Microsoft and the Logitech mouse.

Steps to Reproduce Problem
----

1. Run the Cardfile program that comes with Visual Basic. You'll find it in the Samples subdirectory.

2. Put the mouse cursor on one of the menu labels and press the left mouse button to activate it.

3. While continuing to hold down the left button, move the cursor to a menu item within the pop-up menu to highlight the menu item.

4. While holding the left button down, click the right button once. The menu item should no longer be highlighted.

5. Move the mouse from the item you were selecting. Observe that the mouse no longer activates submenus, and the menu does not retract.

6. Continue to move the mouse down from the menu. At some point, the highlighting of the submenu items will be activated again.

7. Upon stopping on a submenu item and releasing the left button, that menu command will execute.

NOTE: This behavior also occurs if you open a menu and, while holding down the left button, you use the right button to click the screen.

Additional reference words: 1.00 2.00 buglist1.00 buglist2.00 fixlist3.10
KBCategory: kbenv kbbuglist kbfixlist
KBSubcategory: EnvtRun

FIX: Visual Basic List Box Won't Open if Resized at Run Time

Article ID: Q79030

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When you click the down arrow of the drive list box control, the drive list box will not open if it has been resized at run time. The Width property is Read/Write at run time. However, if it is changed at run time, the drive list box won't open. This is true even if it is restored to its original value before attempting to open it.

Note also that Page 11 of the "Microsoft Visual Basic: Language Reference" version 1.0, says the Height property of the drive list box is Read/Write at run time. Height is actually Read-Only at run time.

STATUS

Microsoft has confirmed this to be a bug in the products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Note that neither the directory list box nor the file list box are affected by run-time resizing.

Steps to Reproduce Problem

1. Click the drive list box control icon on the Toolbox. Draw a drive list box on the form. Resize the drive list box to any size. At run time, the drive list box will correctly open when you click the down arrow.

2. Add three command buttons to the form, giving them these captions: Narrow, Wider, and Restore.

3. Insert the following code:

   Note: The example assumes a starting dimension of 2055 wide (user alterable) by 315 high (the standard height in twips).

```
   Sub Command1_Click ()
      drive1.WIDTH = 1025             ' Narrow
   End Sub
   Sub Command2_Click ()
      drive1.WIDTH = 4110             'Wider
   End Sub

   Sub Command3_Click ()
      drive1.WIDTH = 2055             'Restore
   End Sub
```

4. Run the example.

5. Open the drive list box. Click the Narrow or Wider button.

6. Try to open the drive list box again. It fails to open.

7. Click the Restore button. Again try to open the drive list box.
   It fails to open.

Additional reference words: buglist1.00 docerr buglist2.00 fixlist3.00 1.00
2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: SendKeys Causes Erratic Mouse Behavior on IBM PS/2

Article ID: Q79603

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0
- Microsoft Windows version 3.0

SYMPTOMS

When a Visual Basic program executes the SendKeys statement on an IBM PS/2 computer, Windows behaves erratically when you move the mouse until it is shut down.

CAUSE

The erratic behavior is caused by continuous phantom mouse clicks and mouse movements.

STATUS

Microsoft has confirmed this to be a bug in Microsoft Windows version 3.0. This bug was corrected in Microsoft Windows version 3.1.

MORE INFORMATION

If you are running Microsoft Windows 3.0 on a PS/2 computer and you press the NUM LOCK key while moving the mouse, the mouse events become erratic. The Visual Basic SendKeys statement affects the NUM LOCK key, so this problem results -- just as if NUM LOCK were pressed.

When you move the mouse, phantom Click events result in symptoms such as applications unexpectedly launching, or the mouse pointer jumping around the screen.

This problem has been reported to happen on the IBM PS/2 Model 50, Model 50z, Model 60, and Model 80.

Additional reference words: noupd 1.00 3.00 3.10 NUMLOCK
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: File Not Loaded If No Extension in Load Picture Dialog

Article ID: Q80643

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When loading a Windows bitmap, icon, or metafile into the Picture property of a Visual Basic form or picture control by using the Load Picture dialog box (activated by choosing the ellipsis button to the right of the Properties list box), the default filename extension (BMP, WMF, ICO) is not automatically added if you enter a base filename without an extension.

RESOLUTION

Type in the extension at the end of the filename. To correctly load a file, you must specify both the base filename and the extension.

STATUS

Microsoft has confirmed this to be a bug in the products listed above. This bug was corrected in Microsoft Visual Basic version 3.0 for Windows. In Visual Basic version 3.0, .BMP is added as the extension if none is specified.

MORE INFORMATION

Normally, a Windows file list dialog box that displays one or more default extensions will automatically locate and open a file with any of the default extensions if given the base name. However, this does not occur when you select a file for the Picture property by using the Load Picture dialog box.

Steps to Reproduce Problem

1. From the File menu, choose New Project.

2. Select the Picture property from the Properties list box.

3. Choose the ellipsis button on the right of the Settings box to bring up the Load Picture dialog box.

4. Type the name of one of the files listed in the Files box without its extension and choose the OK button.

A "File Not Found" error message displays telling you the selected file was not loaded.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtRun APrgGrap

FIX: Panel Custom Control Caption Not Dimmed When Disabled

Article ID: Q80868

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When the 3-D Panel custom control is disabled by setting the Enabled property to False(0), the controls on the 3-D Panel (if any) are disabled but the caption displayed for the 3-D Panel control is not dimmed.

STATUS

Microsoft has confirmed this to be a bug in the products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Like the standard Frame and Picture controls in Visual Basic, the 3-D Panel can also be used to group other controls together. When these containers or parent controls are disabled by setting the Enabled property to False(0), the controls they contain, or the child controls, are also disabled. In addition, the caption on the frame is dimmed. However, the caption of the 3-D Panel custom control is not dimmed, even though the child controls on it are disabled.

Steps to Reproduce Problem

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

2. From the File menu, choose Add File, and select the THREED.VBX custom control file. The 3-D Panel tool will appear in the Toolbox.

3. Place a panel control (Panel3D1) on the form.

4. Draw a command button (Command1) within the boundaries of the panel control. Make sure the caption of the panel control is still visible.

5. Draw another command button (Command2) on the form outside the panel control, and change its Caption property to Disable panel. This button will be used to disable/enable the panel, which will in turn disable/enable the Command1 button on the panel control.

6. Add the following code to the Command1_Click event procedure:

```
   Sub Command1_Click ()
      MsgBox "You clicked the button!"
   End Sub
```

7. Add the following code to the Command2_Click event procedure:

```
   Sub Command2_Click()
      Panel3D1.Enabled = Not Panel3D1.Enabled 'Enable/Disable panel
         If Panel3D1.Enabled Then
            Command2..Caption = "Disable panel"
         Else
            Command2.Caption = "Enable panel"
      End If
   End Sub
```

8. Press the F5 key to run the program.

With the panel disabled, the Command1 button will produce the message box when clicked. If you click the Command2 button, the Command1 button is disabled as expected. The panel control is disabled but its caption is not dimmed.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00 gray grey grayed greyed
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: Graph Custom Control Incompatible w/ HP II Series Printer

Article ID: Q80912

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS

The Visual Basic Graph custom control cannot successfully print directly to a Hewlett-Packard (HP) II series LaserJet. This is a compatibility issue between the Graph custom control and the HP II series only. It is not a problem with Visual Basic.

WORKAROUND

To work around the problem, add an additional form to the project, transfer the graph's image to the form, and then print the form. This method bypasses the DrawMode=5 (print) method and the incompatibility issue. The example in the More Information section demonstrates how how to implement this workaround.

Note: Unless you know that your graph will never be printed on an HP II Series LaserJet, you may wish to always use this print method.

STATUS

Microsoft has confirmed this to be a bug in the Graph custom control supplied with the products listed above. This bug was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

The Visual Basic Graph custom control version 1.2 allows you to send a graph image directly to your printer by setting the graph's DrawMode property to 5 (print). However, the Graph control is incompatible with the HP II LaserJet family. When using the DrawMode=5 (print) method to print to an HP II LaserJet, only a portion of the graph will print.

Step-by-Step Example

1. Start a new project in Visual Basic. Form1 is created by default.

2. From the File menu, choose Add File, and select the GRAPH.VBX
   custom control file. The Graph tool will appear in the Toolbox.

3. Add another form (Form2).

4. Change the following properties for Form2:

Property	Value
ControlBox	False
MaxButton	False
MinButton	False
Caption	False

5. On Form1, create a Graph control (Graph1) and a command button (Command1). Set the Caption property for Command1 to "Print."

6. Size and edit Graph1 so that it appears the way you want it to print.

7. Add the following code to the Command1_Click event:

```
   Sub Command1_Click ()

      'change to black/white for clearer printing
      Graph1.DrawStyle = 0
      'update change to black/white
      Graph1.DrawMode = 2

      Load Form2
      'size Form2 and transfer Graph1's image
      Form2.width = Graph1.width
      Form2.height = Graph1.height
      Form2.picture = Graph1.picture
      Form2.visible = 1   'optional

      Form2.PrintForm

      'return Graph1 to display in color (optional)
      Graph1.DrawStyle = 1
      'update display to color
      Graph1.DrawMode = 2

   End Sub
```

8. From the Run menu, choose Start (ALT R, S) and click the Print button.

Unless you specify otherwise, Graph1 will originally be displayed in color. Once the Command1_Click event is triggered, the graph will convert to black and white. If you exclude the optional line Form2.visible=1, a dialog box will appear stating that Form2 is being printed. Graph1 will convert back to a color display, and the program will end.

If you included the optional line Form2.visible=1, you will see Form2 appear and resize with the black and white graph image as its picture. A dialog box will appear stating that Form2 is being printed. Graph1 will convert back to a color display and the program will end.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00 H-P HPII
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus APrgPrint

FIX: Animated Button Custom Control: Caption May Be Truncated
Article ID: Q81223

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

The Caption property of the Animated Button custom control allows up to 255 characters to be entered, but only displays a varying number of those characters. The number of characters displayed depends on the FontSize and the width of the characters. The larger the font, or the wider the characters, the less the number of characters that will appear in the caption.

WORKAROUND
==========

To work around the problem, make the font for the caption as small as feasible, and whenever possible, use the smallest size of characters.

STATUS
======

Microsoft has confirmed this to be a bug in the Animated Button custom control provided with the products listed above. This problem was corrected in Animated Button custom control provided with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the ANIBUTTON.VBX custom control file. The Animated Button tool will appear in the toolbox.

3. Place an Animated Button control on Form1.

4. From the Properties list box, select the Caption property for the Animated Button control. Enter 255 W characters. Notice that the caption of the Animated Button control is now filled with W characters.

5. Maximize Form1 by clicking the maximize button in the upper-right corner of Form1.

6. Stretch the right side of the Animated Button control so that you can see the whole caption. (To do this, click the control, and pull the handle on the right side of the control.)

If you change the W characters to I characters, you will be able see more of the characters in the caption before they are truncated because I characters takes proportionately less space than W characters.

Additional reference words: 1.00 2.00 3.00 buglist1.00 buglist2.00 fixlist3.00
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: PrgCtrlsCus

FIX: Gauge: Incomplete Paint with Max-Min Difference > 100

Article ID: Q81462

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS
========

When you use the Gauge custom control, a linear gauge (Style 0) will fail to fill the leftmost column of pixels in the fill area whenever Gauge1.Max - Gauge1.Min is greater than 100. Similarly, the bottom-most row of pixels in the fill area of the horizontal gauge will not be filled given the same condition. The column or row of pixels not filled are cleared to the BackGround color because the inner area is cleared using the BackGround color whenever the Gauge's fill area is updated.

STATUS
======

Microsoft has confirmed this to be a bug in the products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem in Visual Basic Version 1.0
----

1. Start 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. From the File menu, choose Add File (CTRL+F12). In the Files box, select the GAUGE.VBX custom control file. The Gauge tool will appear in the Toolbox.

3. Add the Gauge control to Form1, and set the Gauge's properties to the following:

Properties	Value
BackColor	&H00000000& (Black)
ForeColor	&H00C00C00& (Light Gray)
Max	101
Picture	"SPEEDO.BMP"

   Note that the SPEEDO.BMP is not available in Visual Basic version 2.0.

4. Add the following code to the Gauge_Click event procedure.

```
   Sub Gauge1_Click ()
      For i=Gauge1.Min to Gauge1.Max
         Gauge1.Value = i
      Next i
   End Sub
```

5. From the Run menu, choose Start (ALT, F, N) to run the program.

Note that when you click the Gauge control, there is a black vertical line in the leftmost part of the inner area that isn't filled.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: Graph Custom Control: LabelText May Overlap

Article ID: Q82874

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When you use the Graph custom control, the LabelText strings may overlap. Graph has complete control over the LabelText placement on the x-axis and the size of the font used to display these strings. Each string contained in the LabelText array can be up to 80 characters long. Therefore, depending on the size of the graph and the length of each LabelText string, the labels may overlap on the graph.

STATUS

Microsoft has confirmed this to be a bug in the the Graph custom control shipped with the products listed above. This problem was corrected in the Graph custom control shipped with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the File menu, choose Add File. In the Files box, select the GRAPH.VBX custom control file. The Graph tool appears in the toolbox.

3. Add a Graph control (Graph1) to Form1.

4. Change the following properties for Graph1:

Property	Value	
Top	0	
Left	0	
Width	3500	
Height	2500	
LabelText	aaaaaaaaaaaaaaaaaaaa	(20 a's)
	bbbbbbbbbbbbbbbbbbbb	(20 b's)
	ccccccccccccccccccccccc	(20 c's)
	dddddddddddddddddddd	(20 d's)

```
               eeeeeeeeeeeeeeeee        (17 e's)
```

As you set the properties in step 3, Graph1 will continuously update. Due to the length of the LabelText strings, the labels will stagger themselves on the graph. They can only stagger for three layers before returning to the original level. When you enter the fourth and fifth string (the d's and e's), the labels will overlap with the first and second strings (the a's and b's).

If you reset the Graph1 Width property to 4000, the overlapping disappears.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus APrgGrap

FIX: Graph Custom Control Legends May Print Incorrectly

Article ID: Q82875

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming
  system for Windows, version 1.0

SYMPTOMS

The Graph custom control allows you to place a legend on a graph so that each set of numerical data has a corresponding LegendText property. If one or more of the LegendText properties is a null string, Graph may space the legend texts incorrectly or make them overlap.

WORKAROUND

To work around the problem, add a string of text other than the null string to the LegendText. If you do not want any text to appear, you can add a string of spaces.

Note that this problem is related to your computer's configuration. It can appear worse on some computers, or it may not appear at all. Occasionally, the problem can be circumvented by changing the Width and/or Height property of the graph. However, you cannot calculate the amount necessary to correct the problem, and it may not prove to be a permanent solution. For these reasons we do not suggest this method as a viable workaround.

STATUS

Microsoft has confirmed this to be a bug in the Graph custom control supplied with the products listed above. This problem was corrected in the Graph custom control shipped with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the File menu, choose Add File. In the Files box, select the GRAPH.VBX custom control file. The Graph tool appears in the toolbox.

3. On Form1, add a command button (Command1) and a Graph control (Graph1).

4. Change the following properties:

Control	Property	Value
Command1	Caption	Create Legends
Graph1	NumSets	3
Graph1	Width	3500
Graph1	Height	2100

4. In the Command1 Click event, add the following code:

```
   Sub Command1_Click ()
      Graph1.LegendText = "legend 1"
      Graph1.LegendText = ""
      Graph1.LegendText = "legend 3"
      Graph1.DrawMode = 2       'redraws graph to show new legend texts
   End Sub
```

5. Press F5 to run the program, and click the Command1 button.

When you run the program and click the Command1 button, the graph will update with the three LegendText properties. The second one is a null string and does not appear, but its corresponding colored box does. On most computers, this colored box appears lower than expected, and may be partially overlapped by the legend's third colored box. By changing the Width and/or Height property of Graph1, you can change the placement of the second colored box.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus APrgGrap

FIX: Grid Cell Border May Not Display with Some BackColors

Article ID: Q83759

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

There are some BackColor property values for the Grid custom control that will cause the cell borders to become invisible. The borders of Grid cells are light gray. If you set the BackColor of the Grid to light gray, you will not be able to distinguish the borders from the background of the Grid control.

WORKAROUND
==========

To work around this behavior, you should change the Grid default BackColor(&H00000000&) to a color other than light gray.

STATUS
======

Microsoft has confirmed this to be a bug in the Grid custom control supplied with the products listed above. This problem was corrected in the Grid custom control shipped with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Behavior

There are two ways to lose the outline of Grid cell borders:

- Set the Grid BackColor property, at design time or run time, to light gray (&H00C0C0C0&).
- Set the Windows Background color to gray from the Windows Control Panel. Note that users of your application may encounter this behavior simply by customizing the window colors from the Windows Control Panel.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus APrgGrap

FIX: Toolkit 3-D Option & Check Controls Don't Repaint in 3.1
Article ID: Q84475

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Visual Basic programming system
  for Windows, version 1.0

SYMPTOMS
========

The 3-D Option Button and 3-D Check Box custom controls in the THREED.VBX file do not paint properly if their Value property is changed while the form is loaded (hidden) before being shown. The caption area appears transparent (not painted) until the user clicks it or until the Value is changed in code after the form is shown. This problem occurs only in Windows version 3.1, not Windows version 3.0.

CAUSE
=====

This problem occurs because of changes in the Windows GDI routines to optimize screen refresh performance. For that reason, Windows version 3.1 eliminates what it considers redundant paints.

WORKAROUND
==========

You can work around this problem by assigning the Caption property of the affected controls to themselves when the form is shown again. This code would be placed after the Form2.Show. For example:

```
   Form.Control.Caption = Form.Control.Caption
```

This forces a refresh of the area not being painted. Here are the steps to implement this workaround:

1. Add the following code to the Command2_Click event:

```
   Sub Command2_Click ()
      Option3D3.Value=1
      Check3D3.Value=1
      Form2.Show
      Form2.Option3D1.Caption = Form2.Option3D1.Caption
      Form2.Option3D2.Caption = Form2.Option3D2.Caption
      Form2.Option3D3.Caption = Form2.Option3D3.Caption
      Form2.Check3D1.Caption = Form2.Check3D1.Caption
      Form2.Check3D2.Caption = Form2.Check3D2.Caption
      Form2.Check3D3.Caption = Form2.Check3D3.Caption
   End Sub
```

2. Run the program. Change the values by clicking some checks and options.

3. Click Form2 to hide it.

4. Click the Second Show and notice that the paint is now handled correctly.

You can also work around this problem by explicitly doing a SetFocus call on the control(s) in question. If you are using control array(s), it should be fairly easy. For example, if you had a five-element control array of Check3D1 check boxes, use this code:

```
   Sub Form_Paint()
      For a% = 0 to 4
         Check3D1(a%).SetFocus
      Next
   End Sub
```

RESOLUTION

Sheridan Software, manufacturer of the 3-D Check Box and 3-D Option Button controls, has issued an update to THREED.VBX that corrects the painting problems experienced in Windows version 3.1. To obtain this update, call the Sheridan BBS at (516) 753-5452 (2400 baud) or (516) 753-6510 (9600 baud).

STATUS

Microsoft has confirmed this to be a bug in the products listed above when used in Microsoft Windows version 3.1. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start Visual Basic or from the File menu, choose New Project (ALT, F, N) if Visual Basic is already running. Form1 is created by default. Add a second form (Form2).

2. From the File menu, choose Add File. In the Files box, select the THREED.VBX custom control file. The 3-D tools appear in the Toolbox.

3. Add the following controls to the forms, and set their properties as indicated:

   Form1:

Control	Property	Setting
Form	FormName	Form1
Command button	CtlName	Command1
Command button	Caption	First Show
Command button	CtlName	Command2
Command button	Caption	Second Show

Form2:

Control	Property	Setting
Form	FormName	Form2
3-D Check box	CtlName	Check3D1
3-D Check box	CtlName	Check3D2
3-D Check box	CtlName	Check3D3
3-D Option button	CtlName	Option3D1
3-D Option button	CtlName	Option3D2
3-D Option button	CtlName	Option3D3

4. Add the following code to the Command1_Click event procedure for Form1:

```
Sub Command1_Click
  Form2.Option3D1.Value=1 ' Set values for first show.
  Form2.Check3D1.Value=1
  Form2.Show
End Sub
```

5. Add the following code to the Command2_Click event procedure for Form1:

```
Sub Command2_Click ()
  Form2.Option3D3.Value=1
  Form2.Check3D3.Value=1
  Form2.Show
End Sub
```

6. Add the following code to the Form_Click event procedure for Form2:

```
Sub Form_Click ()
  Form2.Hide
End Sub
```

7. Run the program.

When you click the First Show button, the paint occurs properly for all controls, including the controls whose values were changed in code prior to the show. On Form2, click an option box and a check box to change Values.Click on Form2 to hide the form. Click the Second Show button. The controls whose values changed prior to the form being shown are only painted around the area with the check box or option box. The rest of the area is unpainted.

Reference(s):

Sheridan Software Systems, Inc.
65 Maxess Road
Melville, NY 11747

Phone: (516) 753-0985
Fax: (516) 293-4155

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00 3.10
KBCategory: kbprg kbfixlist kbbuglist

KBSubcategory: PrgCtrlsCus

FIX: Grid Custom Control RemoveItem Does Not Update RowHeight

Article ID: Q85436

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you change the RowHeight property of a Grid control, and then delete a row by using the RemoveItem method, the grid adjusts the height of the rows below the deleted row to their default size. However, it does not update the RowHeight property for those rows. If you reset the RowHeight property to its current value, the Grid does not re-draw the rows to the size given by RowHeight.

WORKAROUND

To work around the problem, set RowHeight to a different value and then change it back to the original value.

For example, replace the code shown in the Command1 Click event in step 6 of the More Information section below with this code:

```
   Sub Command1_Click ()
      For count% = 0 To Grid1.Rows - 1
         Grid1.RowHeight(count%) = 399
      Next count%

      For count% = 0 To Grid1.Rows - 1
         Grid1.RowHeight(count%) = 400
      Next count%
   End Sub
```

STATUS

Microsoft has confirmed this to be a bug in the Grid custom control supplied with the products listed above. This problem was corrected in the Grid custom control shipped with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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. From the File menu, choose Add File. In the Files box, select the
   GRID.VBX custom control file. The Grid tool will appear in the toolbox.

3. Place a Grid control (Grid1) on Form1.

4. Set the Grid1 Rows and Cols properties to 5.

5. Place two command buttons (Command1 and Command2) on Form1.

6. Place the following code in the Command1 Click event:

```
   Sub Command1_Click ()
       For count% = 0 To Grid1.Rows - 1
           Grid1.RowHeight(count%) = 400
       Next count%
   End Sub
```

7. Place the following code in the Command2 Click event:

```
   Sub Command2_Click ()
       Grid1.RemoveItem 1
       For count% = 0 To Grid1.Rows - 1
           Debug.Print Grid1.RowHeight(count%)
       Next count%
   End Sub
```

8. Press F5 to run the program. Click the Command1 button to set
   the RowHeight properties to 400. Click Command2 to remove a row.

   Notice that the grid rows are re-sized even though the output in the
   Immediate window shows that the RowHeight property has not changed.

9. Click Command1. Note that the rows do not re-size.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00
3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: GP Fault or UAE When Unload Form in DragOver Event

Article ID: Q93233

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

If you place an Unload statement in the DragOver event procedure, a general protection (GP) fault or Unrecoverable Application Error (UAE) occurs depending on which version of Windows you are using.

WORKAROUND
==========

Do not place an Unload statement in a DragOver event procedure, or use code to check to make sure that you are done dragging before trying to unload a form. For example, you might use a Timer control. Enable the timer in the DragOver event procedure by setting it to True. Then place the Unload statement in the Timer1_Timer event procedure, and disable the timer by setting the Enabled property to False in the Unload event procedure.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic programming system for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic or from the File menu choose New Project if Visual Basic is already running. Form1 is created by default.

2. Place a Label (Label1) on Form1.

3. Set the DragMode property of Label1 to 1(= Automatic).

4. Add the following code to the Form_DragOver event procedure:

   ```
   Unload Form1
   ```

5. Press the F5 key to run the example and try to drag Label1. A GP fault or UAE occurs.

Additional reference words: 2.00 3.00 GPF buglist2.00 fixlist3.00
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: PrgCtrlsStd PrgCtrlsCus

FIX: UAE/GPF Occurs If EXE Uses Variable Length String in Type

Article ID: Q93256

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

An Unrecoverable Application Error (UAE) or general protection (GP fault can occur in the resulting executable program under the following conditions:

- Your program assigns text to a variable length string
- The variable length string is an element of a user defined type
- You ran the program in the Visual Basic environment immediately before you made the executable version of the program.

The UAE or GP fault occurs when the EXE is run after the VB.EXE environment has been closed. If the project used to generate the EXE is still loaded in the VB.EXE environment, the EXE will run without incident.

WORKAROUND
==========

Load the project and compile it (make the EXE file) without executing the project first. This will create an EXE which will operate without causing the GP fault or UAE.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic programming system for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start VB.EXE.

2. Add Module1.BAS to the project by selecting New Module... from the File menu.

3. Add the following code in Module1.BAS:

```
   Type VBSAMPLE
      X as String
   End Type
   Dim Y as VBSAMPLE
   Sub main ()
      Y.X = "hello"
   End Sub
```

4. From the Options menu, choose Project... Double-click the text 'Form1.' A dropdown box should display the options 'Sub Main' and 'Form1.' Select 'Sub Main.' Then choose the OK button.

5. Press the F5 key to run the example. Then from the File menu, choose Make EXE... Name the executable T1.EXE.

6. Close VB.EXE. (You do not need to save the project, unless you want to use it for future purposes.)

7. From the File Manager, try to run the T1.EXE program. This results in a UAE or GP fault and the loss of the mouse cursor.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: UAE/GPF When Use Static Array in Event Procedure After F5

Article ID: Q93257

```
---------------------------------------------------------------------------
The information in this article applies to:

- The Standard Edition of Microsoft Visual Basic for Windows, version 2.0
---------------------------------------------------------------------------

SYMPTOMS
========

Under very specific conditions, if you add a Static array to an event
procedure, a General Protection Fault (GPF) or Unrecoverable Application
Error (UAE) can occur. This problem is described in further detail below.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard Edition of
Microsoft Visual Basic for Windows, version 2.0. This problem was corrected
in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
--------------------------

1. Start VB.EXE.

2. Press F5 to run the current project and then choose the 'End' option fro
   the 'Run' menu to stop.

3. Add the following line of code to the Form_Load event procedure of
   Form1:

   static F(10) As String ' ** Do not hit the Enter or Return key
                          ' ** to proceed to the next line. Keep
                          ' ** the focus on the same line, you
                          ' ** should not see any blue colored
                          ' ** text, you should only see black text.

4. With the cursor on the same line as the 'static' statement, minimize the
   code window by clicking the minimize arrow of the code window.

5. Press the F5 key to run the project and a GPF or UAE occurs. However,
   if you press the Enter key after typing the line in the Form_Load event
   procedure, you will not encounter this problem. You should now see the
   words 'Static', 'As' and 'String' in the color of blue text. This
   informs you that the line of code has been parsed by the P-code
   interpreter.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes
```

FIX: UAE/GPF When VB Control Name Identical to Property Name

Article ID: Q93424

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  for Windows, version 2.0

SYMPTOMS
========

If you try to change the Name of a control to the same name as an existing Property name, an Unrecoverable Application Error (UAE) or General Protection (GP) fault occurs when you attempt to run the program.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start VB.EXE.

2. Add a combo box and a text box to Form1.

3. Press the F4 key to activate the Properties Window. Select the Name property of the Text1 text box. Change the name from 'Text1' to 'Text'.

4. Add the following code to the Form_Load event procedure of Form1:

   ```
   Forms(0).text.height = 30
   Print Forms(0).combo1.text
   ```

5. Press the F5 key to run the program, and a UAE/GP Fault error occurs.

To prevent this error from occurring, change the Name property back to 'Text1'.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: UAE/GPF When Square Brackets '[]' Around MSGBOX Function

Article ID: Q93425

The information in this article applies to:

- Standard Edition of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS
========

If you enclose an identifier which has a total length of 40 or more characters in square brackets ("[]"), an Unrecoverable Application Error (UAE) or General Protection (GP) fault occurs.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard Edition of Microsoft Visual Basic for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

Square brackets are used when you have an identifier with the same name as a reserved word, and you need to specify that this occurrence of the word is an identifier, and not a use of a reserved word). Identifiers in Visual Basic for Windows, version 2.0 are limited to 40 characters in length.

MORE INFORMATION
================

The following steps can be used to reproduce this problem:

1. Start VB.EXE.

2. Select the New Module routine from the File menu to add Module1.BAS.

3. Add the following to the general declarations section of Module1.BAS:

```
' The following statement should appear on one line.
[BUTTON = MSGBOX("SOME OR ALL OF THE DATES ARE OUT OF RANGE OR
  WERE NOT PROPERLY FORMATTED AND HAVE BEEN RESET TO TODAY'S DATE.
  THE VALID RANGE FOR DATES IS " + RANGE + " AND SHOULD BE IN THE
  FORMA]
```

4. Note the above section of text is on four lines. Place the cursor at the end of the first line and press the Delete key, this appends the second line to the end of the first line. Press the End key to place the cursor at the end of the new first line, and then press the Delete key again, this should append the third line to end of the first line. Proceed on by pressing the End key one more time to place the cursor at the end of the first line. Then press the Delete key again, and this should append the fourth and last line to the first line.

5. With the cursor at the end of new long first line, press the Enter or Return key. A UAE or GPF occurs.

```
   Note: If you try the example above in an event procedure of a procedure
         defined by the user, the same problem occurs.
```

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: GPF/UAE When Converting String > 32K to Double Precision
Article ID: Q93435

The information in this article applies to:

- The Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS
========

When converting a large string that is greater than 32K into a double precision number, a General Protection (GP) fault or Unrecoverable Application Error (UAE) can occur. An example of this problem is described in detail further below. This problem also occurs with the functions CCur, CInt, CLng, CSng as well as CDbl.

WORKAROUND
==========

To work around the problem, break the string into two parts, an x part and a y part. Then you can print both parts one after the other and the error does not occur because each part is less than 32K. Here is an example:

```
   Show
   x = String(20000, "1")
   y = String(20000, "1")
   Print CDbl(x);CDbl(y)
```

STATUS
======

Microsoft has confirmed this to be a problem in the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps can be used to reproduce this problem:

1. Start VB.EXE.

2. Add the following code to the Form_Load event procedure:

```
   Show
   x = String(40000, "1")
   Print CDbl(x)
```

3. Press the F5 key to run the example and a GPF or a UAE occurs.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: VB Painting Problem Occurs When Low on System Resources

Article ID: Q93436

The information in this article applies to:

- The Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0

SYMPTOMS
========

Painting problems can occur if you have a large project with many forms, or you have a small project and are low on system resources at the time the small project is loaded.

CAUSE
=====

These painting problems occur in the Visual Basic programming environment. They usually occur after the environment displays an "Out of Memory" error. This error may mean that your system is very low on system resources and the Visual Basic for Windows programming environment is unable to use any additional system resources to paint the next form or control.

An "Out of Memory" error can occur while coding in the design mode if you place too much code or text into a module or form; there is a limit of 64K of code for each form and module. Or it can occur if you try to add another form or control to a project and are too low on system resources to perform the operation.

WORKAROUND
==========

It is a good idea to stop adding forms, controls, or code once you get this error while the environment is in design mode. You need to close some of your other Windows applications or reduce the number of forms or controls that take up resources. A way to obtain the amount of available resources is to check the About Program Manager... dialog box in the Help menu of the Program Manager. This dialog box displays a percentage of the system resources; if the percentage is getting below 10% or so, you are getting close to receiving an "Out of Memory" error in the Visual Basic for Windows programming environment.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem

----

1. Start VB.EXE.

2. Press the far left button on the Title Bar that contains a picture resembling a form icon and Form2 should display.

3. Repeat step 2 until you get an "Out of Memory" error. Note that the number of forms that can be displayed before getting this error depends on the amount of free system resources that you started with.

4. Once you get this error, try to bring up the Menu Design Windows, by selecting the Window menu. Or try to move the Toolbox, Title Bar or other Windows that are part of the Visual Basic for Windows programming environment. Moving any of these tools may result in a painting problem.

5. Visual Basic for Windows tries to continue working even after the "Out of Memory" error occurs which ends up causing the painting problem. The painting problem occurs because Windows has no resources to give to the Visual Basic for Windows programming environment to perform a repaint of the screen.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Result Differs When Comparing Single w/ Double Precision

Article ID: Q93437

The information in this article applies to:

- The Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0

SYMPTOMS
========

When you compare a real number stored as a Single precision variable to the same real number stored as a Double precision variable, the result may be that they are not equal. Storage of real numbers is different within the two data types. Therefore, the number may be represented differently, so a check for equivalence can return false.

WORKAROUND
==========

In Microsoft Visual Basic version 2.0 for Windows, this problem occurs only on computers that do not have coprocessors. If your computer does not have a coprocessor, add some extra code when comparing data stored in Single data types to those stored in Double data types.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. It was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce the Problem
----

1. Start VB.EXE.

2. Add the following in the Form_Click event procedure of Form1:

```
      Print 12.3! = 12.3#  '** Note the '#' sign disappears.
```

3. Press the F5 key to run the example, and click Form1. If the result is '0' then a coprocessor is not installed, if the result is '-1' then a coprocessor is installed or you are on a 486 with a built in coprocessor.

Note if you run this same example in Microsoft Visual Basic for Windows, version 1.0, the result is '0' with or without a coprocessor installed.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: GPF/UAE When Closing DDE Application from the Task List

Article ID: Q94166

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
=======+

A General Protection (GP) fault or an Unrecoverable Application Error (UAE) occurs under the following conditions:

- A Visual Basic application is actively communicating via a Dynamic Data Exchange (DDE) link
- The Visual Basic application is acting as the destination (or client) in the DDE conversation.
- You close the application by choosing End Task from the Windows task list while the DDE link is still active.

WORKAROUND
==========

To work around the problem, ensure that the DDE conversion terminates before the Visual Basic application terminates by setting the LinkMode property to zero in an event other than the Unload or QueryUnload events for Form1. To do this, you need to enable a timer within the Form_Unload (or Form_QueryUnload) event. Within the Timer event, set the LinkMode property to zero to terminate the DDE conversation.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce this problem:

1. Run Microsoft Excel. Sheet1 is created by default.

2. From the Edit menu, choose Copy to copy cell R1C1 (row 1, column 1) to the clipboard.

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

4. Add a Text box (Text1) to Form1.

5. With the Text box highlighted, choose Paste Link from the Edit menu (ALT, E, L).

6. Type some text in R1C1 in Excel. You should see the result in Text1 of
   Visual Basic.

7. Make an executable program in Visual Basic by choosing Make .EXE File
   from the File menu. Name the executable file P1.EXE.

8. From the Windows Program Manager or Windows File Manager, run P1.EXE.

9. Press CTRL+ESC to bring up the Task List.

10. Select Project1 from the list of programs running.

11. From the Task List, choose the End Task button.

At this point, a GP fault or UAE occurs at address 0011:026A.

To summarize, you can avoid the entire problem by inserting the following steps (4a - 4c) after step 4 shown above. Then redo steps 1 through 11.

4a. Add a timer control (Timer1) to Form1

4b. Add the following code to the Form_Unload event of Form1:

```
   Sub Form_Unload (Cancel As Integer)

      'Cause the DDE conversation to terminate from within the Timer
      'event
         Timer1.Interval = 1
         Timer1.Enabled = True

      'Allow the timer event to occur
         DoEvents

      End Sub
```

4c. Add the following code to the Timer1_Timer event:

```
   Sub Timer1_Timer ()
      'Terminate the DDE conversation
         Text1.LinkMode = 0

      'Timer has served its purpose, so disable it.
         Timer1.Enabled = False
   End Sub
```

Additional reference words: buglist2.00 fixlist3.00 1.00 2.00 3.00 GPF
KBCategory: kbinterop kbprg kbfixlist kbbuglist
KBSubcategory: IAPDDE EnvtRun

FIX: GPF/UAE w/ Stop Command in Event Procedure & Deleted Sub

Article ID: Q94167

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

In Microsoft Visual Basic version 2.0 for Windows, a General Protection (GP) fault or an Unrecoverable Application Error (UAE) occurs when you attempt to delete a Sub or Function when in break mode. This problem does not occur in Microsoft Visual Basic version 1.0 for Windows.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce the problem:

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

2. Enter one line of code into the Form_Load event procedure of Form1:

   ```
   Stop
   ```

3. In the general section enter the following procedure:

   ```
   Sub YourName ()
      '** no code is needed
   End Sub
   ```

4. From the Run menu, choose Start (ALT, R, S). After execution is stopped, go to the YourName procedure, highlight the entire Sub, and then delete it.

5. You will receive this error: "You will have to restart your program after this edit-proceed anyway?" Choose the OK button.

At this point, a GP fault or UAE occurs.

This problem occurs only when you delete the Sub or Function that you were viewing before you ran the program. If you had been viewing the Form_Load event instead of Sub YourName before running the above program, the problem would not have occurred.

Additional reference words: buglist2.00 fixlist3.00 1.00 2.00 3.00 GPF

```
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes
```

FIX: GPF When Pasting 8 Bit .DIB File into Anibutton Control

Article ID: Q94168

The information in this article applies to:

- Standard Edition of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS

If you directly paste a 8 bit color .DIB picture from Paint Brush that comes with Windows, versions 3.0 or 3.1 into the AniButton.VBX control with the Standard Edition of Microsoft Visual Basic for Windows, version 2.0 you may receive a General Protection (GP) fault when using Windows, version 3.1 or an Unrecoverable Application Error (UAE) when using Windows, version 3.0. A GP fault or UAE does not occur if you load the .DIB picture through the Picture property in design mode. Instead, an "Invalid format" error is returned.

STATUS

Microsoft has confirmed this to be a problem in the Standard Edition of Microsoft Visual Basic for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

The following steps can be used to reproduce this problem:

1. Start Paint Brush (Pbrush.EXE) from the Windows subdirectory.

2. Open and load a 8 bit color .DIB file into Paint Brush from the File menu.

3. Click the upper right hand icon that looks like a pair or scissors to cut out the picture of the 8 bit color .DIB file. Cut out a copy of the picture.

4. Select the Copy routine from the Edit menu to make a copy of the clipped area.

5. Start VB.EXE.

6. Select Add File... from the File menu to add the Anibutton.VBX control to the project.

7. Place the Antibutton control on Form1.

8. Press the F4 key to bring up the Properties Window. Select the Frame property and press the '...' button.

9. Once the '...' button is pushed, the 'Select Frame' Window is displayed. Press the Paste key, and an UAE or GPF occurs.

```
Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus
```

FIX: VB MCITEST CD Player Sample Displays Incorrect Track
Article ID: Q94185

The information in this article applies to:

 - The Professional Edition of Microsoft Visual Basic for Windows,
   version 2.0

SYMPTOMS

The CD Player component of the MCITEST.MAK sample program incorrectly displays random numbers in the field labeled "Track."

STATUS

Microsoft has confirmed this to be a bug in the MCITEST sample application supplied with the Professional Edition of Microsoft Visual Basic for Windows, version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

To correct the above problem, add the following code to module GLOBAL.BAS:

```
    Global Const MCI_FORMAT_TMSF = 10
```

Steps to Reproduce Problem

1. Run Visual Basic.

2. From the File menu, select Open Project (press ALT, F, O). Select MCITEST.MAK from the directory VB\SAMPLES\MCI, where "VB" is subdirectory where Visual Basic is located.

3. From the Run menu, select Start.

4. From the Devices menu, select CDAudio, click the Load button, and click the Play button. The track number starts at 1, then incorrectly displays random numbers.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00 multimedia
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: GPF/UAE After Undoing Edit of Option Explicit Statement

Article ID: Q94216

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

A General Protection (GP) fault or Unrecoverable Application Error (UAE) occurs if you attempt to undo the edit of an Option Explicit statement while in break mode.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce this problem:

1. Run Visual Basic, or if Visual Basic is already running, choose New Project from the File menu (ALT, F, N). Form1 is created by default.
2. Under the general declarations section for Form1, enter the statement "Option Explicit" if it is not already there.
3. From the Run menu, choose Start (ALT, R, S)
4. Press CTRL+BREAK to break execution.
5. Edit the Option Explicit statement in the general declarations section of Form1 by pressing the BACKSPACE key to delete the last character from the statement. In other words, delete the t so that you end up with Option Explici.
6. From the Edit menu, choose Undo (ALT, E, U) to undo the edit.
7. Choose the Cancel button when you see this message: "You will have to restart your program after this edit-proceed anyway?"
8. Choose the Cancel button again when you see the same message again: "You will have to restart your program after this edit-proceed anyway?"

At this point, you will experience a GP fault in Windows version 3.1 or a UAE in Windows version 3.0.

Additional reference words: buglist2.00 fixlist3.00 1.00 GPF 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: GPF/UAE When Assign NULL to VBM_GETPROPERTY of type HLSTD

Article ID: Q94217

```
---------------------------------------------------------------------
The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0
---------------------------------------------------------------------

SYMPTOMS
========

When you return a NULL in response to the VBM_GETPROPERTY message for a
custom property of data type HLSTR, a General Protection (GP) fault or
Unrecoverable Application Error (UAE) occurs.

Because of this problem, you cannot simply use the PF_fGetData flag with
a custom property of data type HLSTR. What's more, you must use the
PF_fGetMsg flag to ensure that the value of the property is never set to
NULL. This information is taken from the CDK.TXT file provided with
Microsoft Visual Basic Professional Edition version 2.0 for Windows.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic
Professional Edition version 2.0 for Windows. This problem was corrected
in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Below is the information provided in the CDK.TXT file:

DT-HLSTR Properties and PF_fGetData
-------------------------------------

DT_HLSTR properties cannot use PF_fGetData by itself. They must also
use PF_fGetMsg to avoid returning a NULL hlstr. See the MyTag property
in the PIX example (PIX.C) for a guide for how to properly declare a
HLSTR property and process the VBM_GETPROPERTY message.

Additional reference words: 2.00 3.00 buglist2.00 fixlist3.00
KBCategory: kbtool kbbuglist kbfixlist
KBSubcategory: TlsCDK
```

FIX: Using Graphics Method on DB Objects May Cause GPF/UAE

Article ID: Q94242

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

You may receive a General Protection (GP) fault or Unrecoverable Application Error (UAE) when you try to perform a graphics method on a Database object in Microsoft Visual Basic version 2.0 for Windows.

This problem could happen under many different circumstances. There are eight graphics methods and eight database functions yielding a total of 64 different possible combinations each of which may cause a GP fault or UAE.

CAUSE
=====

The Database functions are not designed to support graphics methods (methods that create graphics in an application). No graphics methods are mentioned in the ODBC section of the "Microsoft Visual Basic Professional Features" manual, version 2.0. However, instead of a GP fault or UAE, you should see an error message such as, "Method not applicable" or error 421.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce the problem:

1. Start VB.EXE.

2. Add the following code to the Form_Click event procedure of Form1:

```
      Dim mydb as Database
      Set mydb = OpenDatabase("NWIND", False, False, Connect)
      mydb.Print "This is a test"
```

3. Press the F5 key to run the example, and click Form1.

The result may be a GP fault or a UAE.

Additional reference words: BugList2.00 fixlist3.00 2.00 3.00 GPF
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: APrgDataODBC

FIX: GPF/UAE When Large Tag w/ MultiSelect of 30+ Controls

Article ID: Q94244

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

If you repeatedly select multiples of several controls on a form, and then attempt to set the Tag property of the controls to a long string, Visual Basic may display an out of memory error followed by a General Protection (GP) fault or Unrecoverable Application Error (UAE).

This problem occurs only in the Visual Basic development environment (VB.EXE), not when running the application.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

To reproduce the problem, repetitively select 30 or more controls and then enter a considerable amount of text for the Tag property. The likelihood of this problem occurring goes up with the number of controls selected and the amount of text assigned to the Tag property.

Steps to Reproduce Problem
----

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

2. Add 30 or more controls to Form1.

3. Select all of the controls on Form1.

4. Assign a large amount of text to the Tag property of the controls on Form1. For example, hold down the A key for about one minute to assign a long string of A characters to the Tag property.

5. Repeat steps 3 and 4 above. Eventually, you will receive an out of memory error. After you get the error, Visual Basic will stop responding (hang) by continually displaying the out of memory message or you will encounter a GP fault or a UAE.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00 GPF
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Setting Add Watch May Cause Your Program to Reset
Article ID: Q94290

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

You may encounter the error message, "Module has changed; must reset" after which the program ends and returns to design mode. Setting a watch point contributes to the problem.

In general, this problem occurs under the following conditions:

- A global, form, or module-level watch point is set.
- The watch point refers to a Visual Basic object such as a form or control.
- The watch point is evaluated from break mode when a modal form is showing.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This bug was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic, or from the File menu, choose New Project (ALT, F, N) if Visual Basic is already running.

2. From the File menu, choose Open Project and open the sample program MDINOTE.MAK. This sample is located in the SAMPLES\MDI directory in your Visual Basic directory.

3. From the Debug menu, choose Add Watch (ALT, D, A).

4. In the Add Watch dialog, enter Forms.Count as the watch expression and choose the OK button.

5. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the sample.

6. From the File menu of the MDINOTE sample program, choose Open (ALT, F, O).

7. From the Visual Basic Debug menu, choose Break (ALT, R, k) or press CTRL+BREAK to break execution.

8. Locate the cmdcancel_Click event procedure within FILEOPEN.FRM. From the Debug menu, choose Toggle breakpoint (ALT, D, T) or press the F9 key to set a break point on the following statement:

   ```
   FileForm.txtFileName.Text = Empty
   ```

9. From the Run menu, choose Continue (ALT, R, C) or press the F5 key to continue running.

10. Choose the Cancel button. Execution stops at the break point you set in step 8.

11. From the Debug menu, choose Single step (ALT, D, S) or press the F8 key to single step.

12. Repeat step 11 until you receive the following error message:

    ```
    Module has changed; must reset
    ```

13. Choose the OK button, and Visual Basic will reset from break mode and return to design mode.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Setting Add Watch May Cause GP Fault or UAE

Article ID: Q94292

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

You may encounter the error message, "Module has changed; must reset" after which the program ends and returns to design mode. Setting a watch point contributes to the problem. In one case, you may receive the error and then get a general protection (GP) fault or an unrecoverable application error (UAE).

In general, this problem occurs under the following conditions:

- A global, form, or module-level watch point is set.
- The watch point refers to a Visual Basic object such as a form or control.
- The watch point is evaluated from break mode when a modal form is showing.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce the problem:

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

2. From the File menu, choose New form (ALT, F, F). Form2 will be created.

3. Add the following code to the Form_Click event procedure of Form1:

```
      Sub Form_Click ()
         Form2.Show 1    '** Show the form modal
      End Sub
```

4. From the Debug menu, choose Add watch (ALT, D, A).

5. From the Add Watch dialog, add the expression Forms.Count, select the Form/Module context option for Form1.Frm, and choose the OK button.

6. Add the following code to the Form_Click event procedure of Form2, and press the F9 key to set a break point on the End Sub

statement.

```
   Sub Form_Click()
      Form2.Hide
   End Sub      '** Set the break point on this line, press F9
```

7. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the sample program.

8. Click in the Form1 form. Form2 will be displayed.

9. From the Debug menu, choose Break (ALT, D, K) or press CTRL+BREAK to break execution. This causes the watch expression to be evaluated.

10. From the Run menu, choose Continue (ALT, R, C) or press the F5 key to continue.

11. Click in the Form2 form. The click event of the Form2 form will cause execution to break.

12. From the Debug menu, choose Single step (ALT, D, S) or press the F8 key to single step.

13. Repeat step 12 until you see the "Module has change; must reset" error message.

14. Choose the OK button.

This results in a GP fault or UAE. In the case of a GP fault, the GP fault normally occurs at address 0001:7F8A in module VB.EXE.

This problem also occurs if you select a global-level watch context in step 5. However, the problem doesn't occur if you select a procedure-level watch context in step 5.

To avoid the problem, either don't set a watch point on an expression that contains a Visual Basic object or don't break execution while a modal form is being shown.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00 GPF
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Painting Problems When FontItalic Set True for Text Box
Article ID: Q94293

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

When you use a text box for input in a program, you will encounter painting problems when the FontItalic property is set to True.

CAUSE
=====

This problem is because of spacing. Italic fonts take up more room for each character entered, but the text box does not account for this. The problem occurs only when you type text into the text box. If you assign text to the Text property at run-time, the problem does not occur.

WORKAROUND
==========

To work around the problem, use the Refresh method to refresh the text box each time a character is pressed. For best results, you should enable a timer from within the KeyPress event for the text box. From within the timer event, you can then use the Refresh method to refresh the contents of the text box.

For example, you can work around the problem by adding the following steps to those listed in the "More Information" section:

6. Add a timer (Timer1) to Form1.

7. Add the following code to the Text1_KeyPress event:

```
   Sub Text1_KeyPress (KeyAscii As Integer)
      Timer1.Interval = 1
      Timer1.Enabled = True
   End Sub
```

8. Add the following code to the Timer1_Timer event for Timer1.

```
   Sub Timer1_Timer ()
      Text1.Refresh

      'Disable the timer since you do not want the timer event
      'to be continually executed
      Timer1.Enabled = False

   End Sub
```

9. From the Run menu, choose Start (ALT, R, S).

10. Enter some text in the Text1 text box. The characters should now
    paint correctly.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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 a text box (Text1) to Form1.

3. Set the FontItalic property to True in the Properties Window.

4. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program.

5. Type ffff (4 f characters) in Text1.

Notice that when you press a character, the previous character does not paint correctly. For example, in the case of using the letter f, only the bottom half of the character paints.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: GPF/UAE When New Project Loaded After Large Previous Proj

Article ID: Q94351

```
---------------------------------------------------------------------
The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0
---------------------------------------------------------------------
```

```
SYMPTOMS
========

A general protection (GP) fault or an unrecoverable application error (UAE)
may occur when you choose New Project from the Files menu and the previous
project loaded had over 3900 procedures. The problem can occur when one
.BAS file has more than 3900 Subs or Functions.

WORKAROUND
==========

To avoid the problem, keep the number of procedures in a single .BAS file
under 3900. Try using more than one .BAS file to hold the 3900 procedures
instead of having all 3900 procedures in one .BAS file.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic
version 2.0 for Windows. This problem was corrected in Microsoft Visual
Basic version 3.0 for Windows.

MORE INFORMATION
================

You may encounter this problem with less then 3900 procedures if lack of
memory is a problem. Each procedure can hold a large amount of code and
create a problem even though you have less than 3900 procedures.

Steps to Reproduce Problem
--------------------------

1. Start VB.EXE.

2. Choose New Module from the File menu.

3. Add the following procedure to MODULE1.BAS (the default module
   name) in the (general) section:

   Sub main ()
      Open "test1.bas" For Output As #1
      For i% = 1 To 4000
         Print #1, "sub sub" + Trim$(Str$(i%))
         Print #1, "end sub"
      Next
      Close
   End Sub
```

4. Choose Project... from the Options menu. In the Project window, select the Start Up Form line and change it from the default Form1 to Sub Main.

5. Press the F5 key or ALT+R+S to run and build the TEST1.BAS file.

6. Choose New Project from the File menu. You don't have to save the project outlined in steps 1 through 5 above.

7. Once a New Project is running, choose Add File... from the File menu.

8. Select the file TEST1.BAS from the Add File window. This is a large (4000 empty procedures) file, so it will take some time to load.

9. Once the TEST1.BAS file is loaded, choose View Code from the Project window with the TEST1.BAS file highlighted. Then you can view the 4000 empty procedures under the (declarations) section.

10. Choose New Project from the File menu. Choose the No button on saving FORM1.FRM, and choose the No button on saving PROJECT1.MAK.

After choosing the second No button, you may receive a UAE or GP fault.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: No Out of Memory Error Generated with Text Box > 32K
Article ID: Q94698

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

The text box in Microsoft Visual Basic version 2.0 for Windows has the capacity to hold up to 32K of text. The problem is that when you try to place more than 32K of text in a text box, no error is generated and no text is added to the text box.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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 a text box (Text1) to Form1.

3. Set the Multiline property of Text1 to True.

4. Set the Scrollbars property of Text1 to 2 - Vertical.

5. Add the following code to the Form_Click event procedure in Form1:

```
   Sub Form_Click ()
      For i% = 1 to 10000
         text1.seltext = Format$(i%, "00000") + Chr$(13) + Chr$(10)
      Next i%
   End Sub
```

6. Press the F5 key to run the procedure and click the Form1 form.

You will see text being added to the text box, but the adding of the text (numbers) stops around the number 04285 and no error is generated. Visual Basic should give you an Out of Memory error message, but it doesn't.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: Attempting to Refresh Null TableDef Field Causes GP Fault

Article ID: Q94773

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

A general protection (GP) fault occurs when you attempt to refresh a Null Fields collection of a TableDef. Instead, you should receive this error: "Method not applicable to this object."

When the Fields collection for a TableDef is not Null, the Refresh method works as expected.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

The following steps reproduce the problem:

1. Start the Professional Edition of VB.EXE with ODBC support already installed.

2. Add the following code to the Form_Click event procedure of Form1:

```
   Form_Click ()
   Dim db As Database
   Dim tDef As TableDef
   DBName$ = "Server1"
   Set db = OpenDatabase(DBName$)   '* DBName$ name of Database
                                    '* that is already setup on
                                    '* the SQL Server. This
                                    '* DBName$ should be set to
                                    '* server name that listed in
                                    '* the ODBC.INI file.
   Set tDef = db.TableDefs(0)

   Set db = OpenDatabase(DBName$)
   tDef.Fields.Refresh        '* This should result in a
                              '* error, but instead results
                              '* in a GP Fault.
   End Sub
```

   In order to reproduce the problem, the first TableDef in the database, TableDefs(0), cannot have any fields associated with it.

3. Press the F5 key or ALT+R+S.

At this point, a GP fault occurs -- usually at address 0008:0083 in VBODBCA.DLL.

To avoid the problem, make sure the Fields collection is not Null before using the Refresh method. To do this, replace the tDef.Fields.Refresh statement in step 2 above with the following code:

```
   If Not tDef.Fields = Null Then
      tDef.Fields.Refresh
   End If
```

Additional reference words: 2.00 3.00 GPF buglist2.00 fixlist3.00
KBCategory: kbinterop kbprg kbbuglist kbfixlist
KBSubcategory: APrgDataODBC

FIX: GPF When Using 8514 Driver with Long String in Text Box

Article ID: Q94774

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

A general protection (GP) fault may occur when you attempt to assign a string longer than 256 characters to a text box. This problem is known to occur when using an ATI Ultra video system with the 8514 Windows video driver.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows when using Windows version 3.1 and the 8514 Windows video driver. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Load the Windows 8514 driver (8514.DRV) by using the Windows Setup program.

2. Start Visual Basic, or from the File menu, choose New Project if Visual Basic is already running. Form1 is created by default.

3. Add a text box (Text1) to Form1.

4. Press the F4 key to select the Properties Window. Set the Multiline property to True and the ScrollBars property to 3 - Both.

5. Add the following code to the Form_Click event procedure of Form1:

```
      For i% = 1 To 100
         text1.SelStart = Len(text1.Text)
         text1.SelText = "This is a test"
      Next i%
```

6. Press the F5 key to run the code.

At this point, you may encounter a GP fault when the length of the string being built in the text box is longer than 256 characters. Note that on some computers, the GP Fault may occur earlier when the total length of the text reaches about 150 characters.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00

```
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd
```

FIX: Changing Decimal Separator Causes Load Errors for Form

Article ID: Q94776

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you change the decimal separator by choosing the International icon from the Windows Control Panel, you can get the error "Errors during load. Refer to ..." when loading a form that was saved as text.

STATUS
======

Microsoft has confirmed this to be a problem with Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

From the Windows Control Panel, you can change the decimal separator for numbers by choosing the International icon. By changing the decimal separator, you can affect the way decimal numbers look when output by statements such as the PRINT method.

However, if you change the decimal separator, you may get the following error when you load a form that had was saved as text (the default): "Errors during load. Refer to" followed by the name of a log file that contains the error information.

When a form is saved as text, all information about the form is saved to the file, this includes all the properties that were changed from their default. Any properties that where written as a decimal number, such as the FontSize property, may not be recognized if you have changed the decimal separator.

Steps to Reproduce Problem
----

1. Start the Windows Control Panel located, by default, in the Main group of the Windows Program Manager.

2. Choose the International icon from the Control Panel to specify international settings.

3. Choose the Change button next to Number Format.

4. Change the decimal separator to a comma (,).

5. Start Microsoft Visual Basic version 2.0.

6. From the File menu, choose Open Project (Press ALT, F, O).

7. Open the Calculator sample program. This is installed in the SAMPLES directory under the CALC subdirectory.

8. From the Project Window, double-click CALC.FRM to load the form.

At this point, you should get the error "Errors during load. Refer to" followed by the name of a log file that contains the error information. Note that although the loading problem was corrected in Visual Basic version 3.0, the Calculator program is not designed ot accept a comma (,) in place of a decimal point. Using a comma causes a "Type mismatch" error.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: GPF When Making .EXE File If Forms Saved as Binary

Article ID: Q94892

The information in this article applies to:

- The Standard Edition of Microsoft Visual Basic for Windows,
  version 2.0.

SYMPTOMS
========

In version 2.0, a general protection (GP) fault can occur when you replace text boxes originally created in version 1.0 with the new version 2.0 Masked Edit Text boxes, and then use the existing code. The GP fault occurs in version 2.0 when you try to turn the Visual Basic project into an executable (.EXE) program.

WORKAROUND
==========

To work around the problem, save the project's forms and code in Text format rather than the default Binary format.

Use the following steps to save the code in Text format:

1. Select each form one by one from the Project window.

2. With each form, go to a code window of that form by double-clicking the form or by pressing the F7 key.

3. From the File menu, choose Save Text... to save each form's code as a .TXT file. Then from the File menu, choose Load Text... and highlight the .TXT file just created; then choose the Replace button.

4. After completing steps 1 through 3 for each form in the project, restart Visual Basic and load the project by choosing Open Project... from the File menu.

5. From the File menu, choose Make Exe...

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Bad .MAK File Prevents Display of Make EXE File Dialog

Article ID: Q94939

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

The Make EXE File dialog box is not displayed and the program is not compiled if you try to make an executable file when the project (.MAK) file was saved incorrectly. Specifically, this problem occurs if the .MAK file was saved with an invalid path to the executable file.

The project file is saved incorrectly if the directory name containing your project files is derived from the Visual Basic working (or current) directory name. This problem occurs if the working directory for Visual Basic has this pattern:

```
   C:\XXXYYY
```

and your project is in a directory that has this pattern:

```
   C:\XXX\TEMP
```

where XXX represents the same pattern of characters.

For example, if you run Visual Basic from a directory called C:\VB2 and your project is in C:\VB\CALC, you will encounter this problem. The Make EXE dialog is not displayed, and your program is not compiled.

WORKAROUND
==========

To work around the problem, use a text editor such as Notepad to delete the line containing "Path=" from your project's .MAK file. Then save the .MAK file, and reload your project in Visual Basic. You will now be able to display the Make EXE dialog box. You will need to delete this statement each time you make an .EXE file.

Another alternative workaround is to place all the files for your project in a new directory where the directory name is not derived from the Visual Basic working directory name. You can then delete the "Path=" statement from the .MAK file using a text editor such as Windows Notepad.

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem

To reproduce this problem, set the working directory of Visual Basic to the directory where VB.EXE is stored (C:\VB). To do this, highlight the Visual Basic icon in Program Manager and choose Properties from the File menu of Program Manager (PROGMAN.EXE). Then in the Properties dialog box, set the working directory path the same path where VB.EXE is located.

1. Create a directory with the same name as the directory where Visual Basic is located excluding the right most character. For example, if Visual Basic is in C:\VB2, create a directory called C:\VB.

2. Create a subdirectory named CALC on the new directory (C:\VB\CALC).

3. Copy all of the files from the Visual Basic SAMPLES\CALC directory to the C:\VB\CALC directory.

4. Start Visual Basic and open the CALC.MAK project in the new C:\VB\CALC directory.

5. From the File menu, choose Make EXE File (ALT, F, K) and choose the OK button to have Visual Basic create an executable using the default name.

6. From the File menu, choose Save Project (ALT, S, V). The project .MAK file will be saved incorrectly. Specifically an invalid relative path such as Path="..2" will be added to the project .MAK file.

7. From the File menu, choose Open Project (ALT, F, O) and open the CALC project.

8. From the File menu, choose Make EXE File (ALT, F, K) and choose the OK button to have Visual Basic create an executable using the default name.

The Make EXE dialog will not be displayed and your program will not be compiled.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: GPF/UAE When Create or Use Huge Array w/ Large Elements

Article ID: Q95290

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

A general protection (GP) fault or unrecoverable application error (UAE) may occur when you try to run or compile a program if an array meets all of the following criteria:

- It is a huge array (greater then 64k in total size).
- The size of the array elements are large (usually 512 bytes or greater). This will usually occur only when the array elements are user-defined type variables.
- An array element contains either one or more variant or variable length string variables.

WORKAROUND
==========

To work around the problem, change the element size of the array elements. In general, the smaller the element size, the less likely the problem.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start VB.EXE.

2. From the File menu, choose New Module (the default is Module1.BAS).

3. Add the following code to Module1.BAS:

```
   Type MyType
      va(1200) As Variant
   End Sub
   Dim ma(20) As MyType
```

4. Press the F5 key to run the code. Then from the Run menu, choose End. At this point a GP fault or UAE may occur.

5. Change the array 'va(1200) As Variant' to 'va(1200) As String'. Note

   that because String variables are 6 bytes and Variants are 16 bytes,
   this change reduces the size of the user-defined type and therefore
   reduces the element size of the array.

6. Press the F5 key to run the code. Then from the Run menu, choose End.

Because you reduced the element size of the array, you may not encounter
a GP fault or UAE this time.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgOther

FIX: Error Message: Timeout While Waiting for DDE Response

Article ID: Q95428

```
---------------------------------------------------------------------
The information in this article applies to:

 - Standard and Professional editions of Microsoft Visual Basic
   programming system for Windows, version 2.0
---------------------------------------------------------------------

SYMPTOMS
========

You can get the error "Timeout while waiting for DDE response" if you
execute DDE commands within a DDE event. This occurs due to a limitation
of the Dynamic Data Exchange Management Library (DDEML.DLL) that provides
support for DDE under Windows. This problem may also occur if you place DDE
commands in an event that is triggered by a DDE command such as the Change
event of a text box.

CAUSE
=====

The problem occurs because changing the text under the Destination Data
section of the DDE source causes a Text1_Change event. Since this is a DDE
related event, attempting to perform a DDE operation such as
text2.LinkRequest results in the timeout error message.

WORKAROUND
==========

To work around the problem, perform all DDE operations in non-DDE
related events. If you need to perform a DDE operation in a DDE related
event, you can put the DDE operations in a timer event that will execute
after the DDE related event has finished. Here is an example:

1. Follow steps 1 through 7 in the More Information section below.

2. Place a timer control (Timer1) on Form1.

3. Set the Interval property of Timer1 to 1.

4. Set the Enabled property of Timer1 to False.

5. In the Text1_Change event, enter the following code:

   Sub Text1_Change ()
      Timer1.Enabled = True
   End Sub

6. In the Timer1_Timer event, enter the following code:

   Sub Timer1_Timer ()
      text2.LinkRequest
      Timer1.Enabled = False
   End Sub
```

7. Run the program.

8. Change the text in the Destination Data section of the compiled DDE
   sample application.

Text1 should correctly display the text typed into the Destination Data
section of the compiled DDE sample application without producing an error.

STATUS

Microsoft has confirmed this to be a problem in the Standard and
Professional editions of Microsoft Visual Basic version 2.0 for Windows.
This problem was corrected in Microsoft Visual Basic version 3.0 for
Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start Visual Basic, or from the File menu, choose New Project
   (ALT, F, N) if Visual Basic is already running.

2. Open the DDE sample program located in the \SAMPLES\DDE directory.

3. From the File menu, choose Make EXE File (ALT, F, K).

4. Start the compiled .EXE program from Program Manager or File Manager.

5. From the File menu, choose New Project (ALT, F, N). Form1 is created
   by default.

6. Place two text boxes on Form1.

7. In the Form_Load event, enter the following code:

```
   Sub Form_Load ()

      text1.LinkMode = 0
      text1.LinkTopic = "dde|system"
      text1.LinkItem = "txtdata"
      text1.LinkMode = 1  'Establish an automatic link

      text2.LinkMode = 0
      text2.LinkTopic = "dde|system"
      text2.LinkItem = "txtdata"
      text2.LinkMode = 2  'Establish a manual link

   End Sub
```

8. In the Text1_Change event, enter the following code:

```
   Sub Text1_Change ()
      text2.LinkRequest
   End Sub
```

9. Run the program.

10. Change the text in the Destination Data section of the compiled DDE
    sample application.

After approximately five seconds, you will receive the error "Timeout
while waiting for DDE response."

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbinterop kbprg kbfixlist kbbuglist
KBSubcategory: IAPDDE

FIX: FixedCols Can Cause Paint Problem with Grid Control

Article ID: Q95429

The information in this article applies to:

- The Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS

Sometimes if the FixedCols property of the GRID.VBX control is set while designing a form, paint problems can occur when the program is run.

WORKAROUND

To work around the problem, set the FixedCols property in code rather than at design time.

STATUS

Microsoft has confirmed this to be a problem in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

This problem appears to happen when the FixedRows property is set to 1 and the FixedCols property is set to something other than 0 or 1. This problem occurs only when you set properties at design time.

Steps to Reproduce Problem

1. Start Visual Basic.

2. From the File menu, choose Add File and add the GRID.VBX custom control.

3. Place a grid control on the form.

4. In the Properties Window, set the Cols property to 3 and the FixedCols property to 2.

5. Run the program.

You should notice some paint problems with the grid control. The grid continues to paint incorrectly until you set the FixedCols property back to 0 or 1 and run the program again.

To avoid the problem, set the FixedCols property at runtime in code:

```
   grid1.FixedCols = 2
```

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: Problems Calling DoEvents from a Scroll Bar Change Event

Article ID: Q95498

```
---------------------------------------------------------------------
The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0
---------------------------------------------------------------------
```

SYMPTOMS
========

Two problems can occur when DoEvents is called from the Change event of a scroll bar.

- Clicking either the up or down directional arrows of a scroll bar causes the change event to fire repeatedly and generate an "Out of Stack Space" error.
- Moving the scroll bar's thumb after clicking either of the directional arrows leads to painting problems with the scroll bar.

WORKAROUND
==========

To work around the problem, move code containing DoEvents calls from the change event to a timer event. Then from the scroll bar change event, enable the timer. For example, add the following steps to those listed in the "More Information" section to implement this workaround:

7. Add a Timer control (Timer1) to Form1.

8. Place the following code in the Timer1_Timer event procedure:

```
   Sub Timer1_Timer ()
      s! = Timer
      Do
      x% = DoEvents ()
      Loop While Timer - s! <= .25
      timer1.Enabled = 0
   End Sub
```

9. Place the following code in the HScroll1 Change event procedure to replace the code added in step 3.

```
   Sub HScroll1_Change ()
      Print "We are in the Change Event"
      timer1.Interval = 2000
      timer1.Enabled = -1
   End Sub
```

10. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program.

Now you should be able to click the directional arrows of the scroll bar and move the scroll thumb without encountering either of the two problems.

STATUS

======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic version 2.0. This problem was corrected in Microsoft Visual Basic version 3.0.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic, or from the File menu, choose New Project if Visual Basic is already running. Form1 is created by default.

2. Add a horizontal scroll bar (HScroll1) to Form1.

3. Add the following code in the HScroll1_Change event procedure of Form1:

```
   Sub HScroll1_Change ()

      Print "We are in the Change Event"

      s! = Timer
      Do
      x% = DoEvents ()
      Loop While Timer - s! <= .25

   End Sub
```

4. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program.

To demonstrate the problem of Change events being fired repeatedly, click either of the scroll bar's directional arrow buttons and leave the mouse cursor over the directional arrow. This will eventually lead to an "Out of stack space" error message.

To demonstrate the painting problems, click either of the arrows. Then move the scroll thumb of the scroll bar in any direction. The scroll bar will be painted incorrectly. This will also lead to an "Out of stack space" error message.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: MAPI: GPF When Attempt to Download 923 or More Messages
Article ID: Q95501

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0

SYMPTOMS
========

The MAPI messages control causes a general protection (GP) fault when you try to download more than 923 messages from the inbox after setting the Action property to MSG_FETCH.

CAUSE
=====

This problem occurs because the internal 64K limit is exceeded when more than 923 messages are fetched into the message set.

WORKAROUND
==========

The only way to avoid this problem is to limit the number of messages downloaded by using the FetchMsgType and FetchUnreadOnly properties to limit the message set to a particular set of messages.

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: Extra Chars in Masked Edit Cause Empty InvalidText Box

Article ID: Q95508

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS
========

Entering more characters than specified in the Mask property of a Masked Edit control generates a ValidationError event, and the InvalidText parameter is set to the empty string.

The InvalidText parameter should be set to the value of the Text property for the masked edit control, including the invalid character.

STATUS
======

Microsoft has confirmed this to be a bug in the Professional Edition of Microsoft Visual Basic version 2.0 for Windows. This problem was partially corrected in Microsoft Visual Basic version 3.0 for Windows. In version 3.0, the InValidText returned has only as many characters as allowed by the Mask. For example, if the Mask is "##" and you type "123" the InvalidText returned is "12"

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start 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. From the File menu, choose Add File. In the Files box, select the MSMASKED.VBX custom control file. The Masked Edit tool appears in the Toolbox. In Visual Basic version 3.0, MSMASKED.VBX is automatically installed.

3. Add a masked edit control (MaskedEdit1) to Form1 and change its Mask property to ##.

4. Add the following code to MaskedEdit1_ValidationError:

```
   Sub MaskedEdit1_ValidationError (InvalidText As String,
      StartPosition As Integer) 'This must be on a single line.
      MsgBox InvalidText
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) to run the program.

6. Type 123 into the masked edit control.

At this point, you'll see an empty message box. Instead of being empty, the message box should display "12" -- the masked portion of the "123" entered in step 6.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00 blank
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: Text Box/Mask Edit in Select Mode If MsgBox in LostFocus
Article ID: Q95509

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

If you use the mouse to click a text box or a Masked Edit control moving the focus off a control that executes a MsgBox statement in its LostFocus or ValidateError event, the insertion point goes into select mode once the message box is closed. After closing the message box, if you move the mouse cursor from side to side of the Text Box or Masked Edit control, text in the control is selected based on the point where the mouse was clicked to move focus to the Text Box or Masked Edit control. Clicking the mouse anywhere within the Text Box or Masked Edit control turns off select mode.

STATUS
======

Microsoft has confirmed this to be a problem in the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows and in Microsoft Visual Basic programming system version 1.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

Steps to Reproduce Problem
----

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 a text box (Text1) to Form1.

3. Add a command button (Command1) to Form1.

4. Add the following code to Command1_LostFocus.

```
   Sub Command1_LostFocus
      MsgBox "Command1 LostFocus Event"
   End Sub
```

5. From the Run menu, choose Start (ALT, R, S) to run the program.

6. Click the Command1 button to bring the focus to it.

7. Click the x in Text1 in the text box. The message box appears. Click the OK button to Close the message box.

8. Move the mouse cursor over the word Text1 in the text box and then move it left or right.

When you move the mouse cursor from side to side of the Text Box, you select the text on either side of the x in Text1. The insertion point should not select text; it should only represent the entry point for any text entered.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus PrgCtrlsStd

FIX: Focus Rectangle Remains When Grid Loses Focus
Article ID: Q95514

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0

SYMPTOMS
========

When a grid control loses focus, the focus rectangle surrounding the active cell incorrectly remains on the cell.

This behavior differs from that of the grid control that shipped with the Professional Toolkit for Visual Basic version 1.0. The active cell, with the focus rectangle, can be differentiated from other cells in the grid by its wider border when GridLines is set to True or by the fact that it is the only cell with a border when GridLines is set to False.

WORKAROUND
==========

To work around the problem, change the active cell to one in a fixed row or column so that no cell has a focus rectangle. Selected cells are unaffected by changing the active cell. For example, add the following code to the LostFocus event of a grid control named grid1:

```
   Sub Grid1_LostFocus ()
      Grid1.Row = 0
      Grid1.Col = 0
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a problem in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

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. From the File menu, choose Add File. In the Files box, select the GRID.VBX custom control file. The grid tool appears in the Toolbox. In Visual Basic version 3.0, GRID.VBX is automatically installed.

3. Add a grid control (Grid1) to Form1 with the following properties:

   Rows: 5
   Cols: 5

4. Add a text box (Text1) to Form1.

5. From the Run menu, choose Start (ALT, R, S) to run the program. Grid1 gets the focus on startup and the focus rectangle is around R1C1.

6. Tab to the text box. Focus changes from Grid1 to Text1. Even though focus changed to Text1, the focus rectangle on R1C1 on the Grid1 incorrectly remains.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: GPF When Erase User-Defined Array of Variable Strings

Article ID: Q95525

```
----------------------------------------------------------------------
The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 2.0
----------------------------------------------------------------------
```

SYMPTOMS
========

If you try to erase an user-defined type array of a variable-length strings, you may encounter a general protection (GP) fault or unrecoverable application error (UAE).

WORKAROUND
==========

This problem doesn't occur if you use an array of fixed-length strings or an array of type Variant in place of the array of variable-length strings. Therefore, you can work around the problem by using an array such as the following with a user-defined type and fixed-length strings.

```
   Type mytype
      mystrings(1) As String * 10   'array of fixed length string
   End Type

   Global test As mytype
```

You can also work around the problem by using an array of variants instead of an array of strings, as this example shows:

```
   Type mytype
      mystrings(1) As Variant      'array of variant type
   End Type

   Global test As mytype
```

A third alternative is to erase the elements in the variable-length string array manually instead of using the Erase statement, as follows:

```
   Form_Click()
   For i% = 0 to UBound(test.mystrings)
      test.mystrings(i%) = ""
   Next i%
   End Sub
```

STATUS
======

Microsoft has confirmed this to be a problem in Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

================

Steps to Reproduce Problem
----

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

2. From the File menu, choose New Module (ALT, F, M). Module1 will be
   created.

3. Add the following code to the general declarations section of Module1:

```
      Type mytype
         mystrings(1) As String
      End Type

      Global test As mytype
```

3. Next add the following code to the Form_Click event procedure of Form1:

```
      Form_Click()
         Erase test.mystrings(1)
      End Sub
```

4. Press the F5 key and click Form1.

At this point, you will encounter a GP fault or UAE.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgOptTips

FIX: Loading Proj Gives Err: Custom control 'Graph' not found

Article ID: Q95590

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS
========

When loading a project in Visual Basic for Windows, you see the following error message:

```
   Custom control 'Graph' not found
```

Or an unrecoverable application error (UAE) in Windows version 3.0 or a general protection (GP) fault in VBRUN100.DLL at 0058:0485 in Windows version 3.1 may occur as a result of an executable (.EXE) file created in Visual Basic version 1.0.

CAUSE
=====

If you have the Professional Toolkit for Visual Basic version 1.0 for Windows installed on your computer and you install the Professional Edition of Visual Basic version 2.0 for Windows on the same computer, the new installation may replace the version 1.0 GRAPH.VBX, GSWDLL.DLL, and GSW.EXE files with the Professional Edition of Visual Basic version 2.0 for Windows files.

WORKAROUND
==========

To work around the problem, replace any version 1.0 Graph controls on your forms with the new version 2.0 Graph controls, or re-install the earlier versions of GRAPH.VBX, GSWDLL.DLL, and GSW.EXE. For the best results, you should upgrade the entire project to Visual Basic version 2.0, and then use the newer version 2.0 controls.

STATUS
======

Microsoft has confirmed this to be a bug in the products listed above. This problem was corrected in the Graph control provided with Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start Visual Basic version 1.0.

2. From the File menu, choose Add File and add the version 1.0 graph
   control (GRAPH.VBX) to the project.

3. Choose the Graph icon from the toolbox and draw a graph on the form.

4. Save all changes and exit Visual Basic.

5. Replace the files GRAPH.VBX, GSWDLL.DLL, and GSW.EXE with the new
   version 2.0 files.

6. Start Visual Basic and load the project you created.

You will get the error "Custom control 'Graph' not found." To work
around this problem, replace any version 1.0 Graph controls on your
forms with the new version 2.0 Graph controls or re-install the
earlier versions of GRAPH.VBX, GSWDLL.DLL, and GSW.EXE.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00
3.00 GPF errmsg
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtDes

FIX: Resizing MDIForm with UI Does Not Update Height & Width

Article ID: Q96097

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0

SYMPTOMS
========

If a user resizes a MDIForm at run time with the mouse, the Height and Width properties for the MDIForm incorrectly retain their previous values. Resizing the form by using the user interface (the mouse or the system menus) should change the Height and Width properties to reflect the new size of the MDIForm. Changing the Height and Width properties in code does correctly update the properties.

WORKAROUND
==========

This problem occurs only when a user uses the user interface to change the size of the MDIForm. Therefore, to work around the problem, you can use code to change the Width and Height properties.

The GetWindowRect Windows API function retrieves the dimensions of the bounding rectangle of a given window, including the title bar, border, and scroll bars, if present. You can use the GetWindowRect, to update the Width and Height properties in a program as the properties change in the Resize event of the MDIForm.

The following example demonstrates this workaround:

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. Set the MDIChild property to True for Form1.

3. From the File menu, choose New Module (ALT, F, M). Module1 is created by default.

4. Add the following code to the General Declarations section of Module1:

```
   Declare Sub GetWindowRect Lib "USER.EXE" (ByVal h%, rect As Any)
   Type RectShort
      X As Integer
      y As Integer
      dx As Integer
      dy As Integer
   End Type
```

5. From the File menu, choose New MDI Form (ALT, F, I). MDIForm1 is created by default.

6. Add the following code to MDIForm1's MDIForm_Resize event procedure:

```
   Sub MDIForm_Resize ()
      Dim rect As RectShort

      Call GetWindowRect(Me.hWnd, rect)

      If (rect.dx - rect.X) * Screen.TwipsPerPixelX <> Width Then
         Me.Width = (rect.dx - rect.X) * Screen.TwipsPerPixelX
      End If

      If (rect.dy - rect.y) * Screen.TwipsPerPixelY <> Height Then
         Me.Height = (rect.dy - rect.y) * Screen.TwipsPerPixelY
      End If

   End Sub
```

7. Add the following code to the Form1_Click event:

```
   Sub Form1_Click ()
      Print "Width = "; Format$(MDIForm1.Width)
      Print "Height = "; Format$(MDIForm1.Height)
   End Sub
```

8. From the Run menu, choose Start (ALT, R, S) to run the program.

9. Using the mouse, grab the lower right-hand corner border of MDIForm1.
   Resize it so that the MDIform is taller and wider than its current size.

10. Click the command button.

At this point, the current Height and Width properties for MDIForm1 are printed on Form1.

11. Repeat steps 9 and 10.

The current Height and Width properties for MDIForm1 are printed on Form1 reflecting their new values.

STATUS

Microsoft has confirmed this to be a problem in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

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. Set the MDIChild property to True for Form1.

3. Add the following code to Form1's Form_Click event procedure.

```
   Sub Form_Click ()
      Print mdiform1.Width, mdiform1.Height
      Print mdiform1.ScaleWidth, mdiform1.ScaleHeight
   End Sub
```

4. From the File menu, choose New MDI Form (ALT, F, I). MDIForm1 is created by default.

5. From the Run menu, choose Start (ALT, R, S) to run the program.

6. Use the mouse and click Form1.

The Width and Height property values for Form1 are printed on the first line of Form1, and its ScaleHeight and ScaleWidth are printed on the second line.

7. Use the mouse to grab the lower right-hand corner border of MDIForm1. Resize it so that the MDIform is taller and wider than the default size it had originally.

8. Using the mouse, click Form1.

The Width and Height property values for Form1 are printed on the third line of Form1, and its ScaleHeight and ScaleWidth are printed on the fourth line.

As expected, the ScaleHeight and ScaleWidth values on the fourth line are larger than their corresponding values on the second line. The Width and Height properties on line three, however, are identical with line one. Like the ScaleHeight and ScaleWidth, the Height and Width values should change reflecting the form's new size.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: Scroll Bar Thumb Doesn't Do Change Event as It Should

Article ID: Q96798

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0

SYMPTOMS

A Change event is generated when Visual Basic code sets a scroll bar's Value property. However, if the user then drags the thumb (scroll box) on the scroll bar to either its minimum or maximum value, a change event should occur but may not. The change event is generated correctly when the thumb on the scroll bar is dragged to any point other then its minimum or maximum after Visual Basic code sets the Value property.

STATUS

Microsoft has confirmed this to be a bug in the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. Start 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 a horizontal scroll bar (HScroll1) to Form1.

3. Add a label (Label1) to Form1.

4. Add a command button (Command1) to Form1.

5. Add the following code to Form1's Form_Load event procedure:

```
   Sub Form_Load ()
      Form1.Show
      HScroll1.Value = 1
      HScroll1.Min = 1
      HScroll1.Max = 100
   End Sub
```

6. Add the following code to the Command1_Click event procedure:

```
   Sub Command1_Click ()
      HScroll1.Value = HScroll1.Max
   End Sub
```

7. Add the following code to the HScroll1_Change event procedure:

```
   Sub HScroll1_Change ()
      Label1.Caption = Str$(HScroll1.Value)
   End Sub
```

8. From the Run menu, choose Start (ALT, R, S) to run the program.

9. Choose the command button. The thumb on the scroll bar correctly moves to its maximum position and the label displays the Max property of HScroll1, 100.

10. Drag and drop the thumb on the scroll bar back to its minimum position. The label incorrectly continues to display the Max property for HScroll1, 100. A change event should have occurred in HScroll1 when the thumb was dragged back to its minimum position, and the caption should have changed to 1. But the change event was not generated.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtRun

FIX: Can't Open ODBCADM.HLP Err Msg During Data Access Setup

Article ID: Q97083

The information in this article applies to:

- The Professional Edition of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS

While setting up data access, you receive this error message:

```
   Unable to open the file ODBCADM.HLP
```

This occurs while running the Data Access Setup program either upon completion of setting up Visual Basic version 2.0 or later by choosing the icon created in Program Manager by the Visual Basic setup program and then choosing to install ODBC in the VBDIR\ODBC directory. The VBDIR in VBDIR\ODBC is the directory (default C:\VB) where you installed Visual Basic.

WORKAROUND

Choose one of the following to work around the problem:

- Install ODBC in a directory other than VBDIR\ODBC.
- While installing data access, choose not to install the ODBC Administration Utility.

The ODBC Administration Utility is recommended for managing the data sources for ODBC, so installing ODBC in a directory other than VBDIR\ODBC is the best of the two alternatives.

STATUS

Microsoft has confirmed this to be a bug in the Professional Edition of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce the Problem

1. Run the Visual Basic 2.0 Professional Edition Setup program.

2. Select C:\VB as directory to install Visual Basic.

3. Select Option to Install Data Access.

4. Select C:\VB\ODBC as the destination directory for ODBC.

Midway through copying the files over, the Data Access Setup program displays the following error message and you are forced to cancel setup:

```
   Unable to open the file ODBCADM.HLP. It is in use by another application
```

Additional reference words: 2.00 3.00 buglist2.00 fixlist3.00
KBCategory: kbinterop kbprg kbbuglist kbfixlist
KBSubcategory: APrgDataODBC

FIX: No Menu Event with Maximized MDI Child

Article ID: Q97135

```
----------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 2.0
----------------------------------------------------------------------
```

SYMPTOMS
========

A top level menu's click event on an MDI form isn't fired as it should be when the MDI child is maximized and a sub-menu item exists for that top level menu. There is no click event generated regardless of whether the menu is part of the MDI child or the MDI parent.

STATUS
======

Microsoft has confirmed this to be a bug in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
----

1. Start VB.EXE.

2. Change the MDIChild property of Form1 to True.

3. From the File menu, choose New MDI Form (ALT+F+I)

4. From the Window menu, choose Menu Design (ALT+W+M), and add two menu
   items. Indent the second item once.

Caption	Name
&Top Level	mTopLevel
&SubMenu	mTopLevelSubMenu

5. Add the following code to their respective event procedures:

```
   Sub mTopLevel_Click ()
      Form1.Print "TopLevel"
   End Sub

   Sub mTopLevelSubMenu_Click ()
      Form1.Print "SubMenu"
   End Sub

   Sub MDIForm_Load ()
```

```
      Form1.Show
   End Sub
```

6. From the Run menu, choose Start.

7. Select the Top Level menu item to see a message printed on Form1.

8. Maximize Form1 and Select the Top Level menu item. A message should be printed but is not.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: Mouse Misbehaves After Changing Graph Visible Property

Article ID: Q97588

```
---------------------------------------------------------------------
The information in this article applies to:

- The Professional Edition of Microsoft Visual Basic for Windows,
  version 2.0
---------------------------------------------------------------------

SYMPTOMS
========

If you set the Visible property of a graph control (GRAPH.VBX) to False
from the Change event of a scroll bar, the mouse behaves as if its
button is being held down even after you release it.

STATUS
======

Microsoft has confirmed this to be a bug in the Professional Edition of
Microsoft Visual Basic version 2.0 for Windows. This problem was
corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION
================

Steps to Reproduce Problem
--------------------------

1. Start 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. From the File menu, choose Add File. In the Files box, select the
   GRAPH.VBX custom control file.

3. Place a horizontal scroll bar named HScroll1 on Form1. Set the Maximum
   property to 3.

4. Add four graphs to Form1 using the same name for each one to make it
   into a control array.

5. Add the following code to the Form_Load event:

   Sub Form_Load ()
      Graph1(0).Visible = True
      Graph1(1).Visible = False
      Graph1(2).Visible = False
      Graph1(3).Visible = False
   End Sub

6. Add the following code to the HScroll1_Change event:

   Sub HScroll1_Change ()
      For i = 0 To 3
         ' Set graph Visible property to true if i matches scroll var value
         ' otherwise to false.
         Graph1(i).Visible = (i = HScroll1.Value)
      Next
   End Sub
```

7. Run the program. Click the scroll bar right arrow without moving the
   mouse pointer away. Instead of displaying the next graph control in the
   control array, the program incorrectly scrolls through all the graph
   controls leaving the scroll bar at its maximum value.

Additional reference words: buglist2.00 fixlist3.00 2.00 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsCus

FIX: OLE Client: Copying Linked Object Gives Err: Can't Paste

Article ID: Q97619

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 2.0
- Microsoft Professional Toolkit for Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you use the OLE client control to paste a linked OLE object onto the clipboard and then later copy the same OLE object from the clipboard back to the OLE client control, you may see this error message:

```
   Can't Paste
```

This occurs whether the linked OLE object is created from an existing file (OleClient1.Action = 1) or from an OLE object on the clipboard (OleClient1.Action = 4). This problem occurs only with a linked object, not with an embedded object.

STATUS

Microsoft has confirmed this to be a bug in both the Standard and Professional Editions of Microsoft Visual Basic version 2.0 for Windows and in the Microsoft Professional Toolkit for Visual Basic programming system version 1.0 for Windows. This bug was corrected in Visual Basic version 3.0 for Windows.

MORE INFORMATION

The following example uses Microsoft Excel version 4.0 as the application associated with the OLE object, however the bug does not depend on Excel; it occurs no matter which application is associated with the OLE object.

Steps to Reproduce Problem

1. Start Microsoft Excel. The Sheet1 worksheet is created by default.

2. In the R1C1 cell, enter Fixed Assets.

3. From the Edit menu, choose Copy (ALT+E+C).

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

5. From the File menu, choose Add File. In the Files box, select the OLECLIEN.VBX custom control file. The OLE client tool appears in the Toolbox.

6. Add an OLE client control (OleClient1) to Form1.

7. Add a command button (Command1) to Form1.

8. Add the following code to the Command1_Click event:

```
   Sub Command1_Click ()
      Const OLE_LINKED = 0
      Const OLE_COPY = 4
      Const OLE_PASTE = 5
      Const OLE_UPDATE = 6
      Const OLE_DELETE = 10

      If OleClient1.PasteOK Then
         OleClient1.Protocol = "StdFileEditing"
         OleClient1.ServerType = OLE_LINKED
         OleClient1.Action = OLE_PASTE  ' Get object from clipboard
         OleClient1.Action = OLE_COPY   ' Copy the object back onto the
                                        ' clipboard
         OleClient1.Action = OLE_UPDATE ' Display object
         OleClient1.Action = OLE_PASTE  ' Attempt to paste the
                                        ' object onto the clipboard
         OleClient1.Action = OLE_DELETE
      Else
         MsgBox "Contents of the Clipboard in unacceptable format"
      End If
   End Sub
```

9. From the Run menu, choose Start (ALT+R+S) to run the program.

10. Click the Command1 button. It should work, but instead the program stops and gives the "Can't paste" error message. The Excel object is successfully linked to OleClient1 and displayed, and the linked object is also copied successfully onto the clipboard.

Additional reference words: 1.00 2.00 3.00 errmsg buglist1.00 buglist2.00 fixlist3.00
KBCategory: kbole kbbuglist kbfixlist
KBSubcategory: IAPOLE

FIX: GPF/UAE with Huge Array Size as Multiple of 64K Bytes

Article ID: Q98990

The information in this article applies to:

- Standard and professional editions of Microsoft Visual Basic programming
  system for Windows, version 2.0

SYMPTOMS

A general protection (GP) fault or Unrecoverable Application Error (UAE)
may result when you define a huge array using DIM, REDIM, or GLOBAL and
specify a size that's a multiple of 64K.

CAUSE

Huge arrays that cause a GP fault or UAE are a(n), where n is 4094 + 4095*i
for i = 1 to 7 (assuming 16-byte element sizes). The problem occurs when
the array plus its overhead fills a space of 128K and each increment of 64K
exactly.

WORKAROUND

To work around the problem, add or subtract one element in the array.

STATUS

Microsoft has confirmed this to be a bug in Microsoft Visual Basic
version 2.0 for Windows. This bug was corrected in Microsoft Visual
Basic version 3.0 for Windows.

MORE INFORMATION

Steps to Reproduce the Problem

1. Start Visual Basic, or if Visual Basic is already running, choose New
   Project from the File menu (ALT, F, N).

2. Add the following code to the Form_Click event procedure of Form1:

```
   Form_Click ()
       ReDim A(32759) As Variant
   End Sub
```

3. From the Run menu, choose Start (ALT, R, S).

At this point, a GP fault or UAE occurs. The GP fault address is 0001:0CA2.

Additional reference words: buglist2.00 fixlist3.00 2.00
KBCategory: kbenv kbfixlist kbbuglist

KBSubcategory: EnvtRun

FIX: Erase Won't Clear Contents of Huge Fixed Array as Variant

Article ID: Q99457

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic programming
  system for Windows, versions 2.0 and 3.0

SYMPTOMS
========

The Erase statement fails to erase huge static arrays of type Variant. This
problem occurs with the Variant data type only.

The problem does not occur if the size of the array is less than 64K or if
you use a huge dynamic array of type Variant.

CAUSE
=====

This problem occurs with huge static arrays of the variant data type. An
array is static when you dimension it with the Static keyword or if you
use the DIM keyword to dimension the array in the general-declaration
section of a form or module.

The problem occurs because the Erase statement corrupts the array
descriptor for a huge static array of variants. However, only the
references to the 64K data segments other than the first segment are
corrupted. Any elements in the first 64K segment of the array are always
erased properly. All elements stored in other segments are not erased.

The Erase statement is only effective the first time you erase the elements
of a huge static variant array. Any additional attempt to Erase elements
of
the array will fail and the elements in the array in data segments other
than the first segment will not be erased.

WORKAROUND
==========

To work around the problem, clear each element of the array manually by
setting each element to Empty. Replace the "Erase a" statement in step 2
shown below with this code:

```
   For i% = 0 to 5000
      a(i%) = Empty            '** Empty = 0
   Next i%
```

STATUS
======

Microsoft has confirmed this to be a bug in Microsoft Visual Basic
version 2.0 for Windows. This problem was corrected in Microsoft Visual
Basic version 3.0 for Windows.

```
MORE INFORMATION
================

Steps to Reproduce Problem
--------------------------

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

2. Enter the following procedure into the general section of Form1:

   Sub test()

     Static a(5000) As Variant  'Huge static variant array

     a(1) = 1              '*element 1 is in the first segment
     a(100) = 2            '*element 100 is in the first segment
     a(5000) = 3           '*element 5000 is in the second segment

     Debug.Print "Before the Erase:"
     Debug.Print "a(1) = "; a(1)
     Debug.Print "a(100) = "; a(100)
     Debug.Print "a(5000) = "; a(5000)
     Debug.Print ""

     Erase a               '*erase the elements

     Debug.Print "After the Erase:"
     Debug.Print "a(1) = "; a(1)
     Debug.Print "a(100) = "; a(100)
     Debug.Print "a(5000) = "; a(5000)
     Debug.Print ""

   End Sub

3. Place the following code in the Form_Click event procedure for Form1:

   Form_Click ()
      Call test
   End Sub

4. Press F5 to run the example. Click Form1 to see the following results
   in the Debug Window:

   Before the Erase:
   a(1) =  1
   a(100) =  2
   a(5000) =  3

   After the Erase:
   a(1) =
   a(100) =
   a(5000) =

   But if you click again, you will see different results:

   Before the Erase
   a(1) =  1
   a(100) =  2
   a(5000) =  3

   After the Erase
   a(1) =
   a(100) =
   a(5000) =  3
```

This shows that the elements of the huge static Variant array were not cleared, but the elements of a smaller Variant array were cleared.

Additional reference words: buglist2.00 fixlist3.00 1.00
KBCategory: kbenv kbfixlist kbbuglist
KBSubcategory: EnvtRun

FIX: VB 2.0 Prof Demo Causes Error: Invalid File Format
Article ID: Q100611

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 2.0

SYMPTOMS

Running the Professional Edition Demo for Visual Basic version 2.0 for Windows may immediately cause an "Invalid File Format" error. This problem may also occur when you run a Visual Basic program that uses the common dialog custom control.

CAUSE

This error is usually caused by an incorrect version of the common dialog VBX (CMDIALOG.VBX) file in the \WINDOWS directory. The Visual Basic version 1.0 Professional Toolkit installs the common dialog VBX into the \WINDOWS directory, whereas the version 2.0 Professional Edition installs the common dialog VBX into the \WINDOWS\SYSTEM directory leaving the old version of the common dialog VBX in the \WINDOWS directory.

When the version 2.0 professional demo is run, the demo finds the old common dialog VBX in the Windows directory first and gives the error "Invalid File Format."

WORKAROUND

To work around the problem, delete or move the Visual Basic version 1.0 version of the CMDIALOG.VBX out of the \WINDOWS directory. This will leave the correct version of the common dialog VBX in the \WINDOWS\SYSTEM directory.

STATUS

Microsoft has confirmed this to be a bug in the product listed above. This bug was corrected in Microsoft Visual Basic version 3.0 for Windows. In Visual Basic version 3.0, the Common dialog control ships with both the Standard and Professional editions, so the version 3.0 Professional edition demo doesn't discuss the Common dialog control, which avoids the the error. Version 3.0 of the Common dialog control replaces version 2.0 of the control.

Additional reference words: buglist2.00 fixlist3.00 2.00 errmsg 3.00
KBCategory: kbprg kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

FIX: Repaint Prob Adding Graphical Control as Child of Graph

Article ID: Q102606

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, version 2.0

SYMPTOMS

In Visual Basic version 2.0 if you add one of the graphical controls (label, image, or line control) as a child to the graph control, the graph image within the graph control is repainted incorrectly and the graphical control moved behind the graph image. The problem does not occur when non-graphical controls are used.

CAUSE

This is caused by a bug in the graph control where the repainting of the graph image cannot handle the graphical controls as child controls.

WORKAROUND

To avoid this problem, place a picture box as a child on the graph. Then place the graphical control in the picture box. This works well when using the label control but is not very useful when using the other graphical controls.

The only other way to work around this problem in Visual Basic version 2.0 is to not add a graphical control as a child of the Graph control; that is, use only non-graphical controls.

STATUS

Microsoft has confirmed this to be a bug in Visual Basic version 2.0 for Windows. This problem was corrected in Visual Basic version 3.0 for Windows

MORE INFORMATION

This problem was fixed in Visual Basic version 3.0 with the new version of the graph control (GRAPH.VBX version 2.0). The solution was to remove the ability of the graph control to support child controls. Therefore, in Visual Basic version 3.0, you cannot add any control as a child to the graph control.

Additional reference words: 2.00 3.00 buglist2.00 fixlist3.00
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: PrgCtrlsCus

FIX: Invalid Argument Err on Execute Method w/ SQL Passthrough

Article ID: Q103976

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

When you open a database using ODBC and use the Execute method of the Database object or property with the SQL passthrough option (value 64) specified, the error "Invalid argument" (number 3001) incorrectly occurs.

WORKAROUND
==========

Here are two possible workarounds. Use either one.

- Use the ExecuteSQL. Its default is DB_SQLPASSTHROUGH:

```
      i = db.ExecuteSQL("action statement")
```

- Use CreateDynaset or CreateSnapshot with the SQL passthrough option to execute an SQL action statement. Then close the resulting recordset object immediately. Here's an example:

```
      Dim ds As Dynaset
      Set ds = db.CreateDynaset("action statement", SQL_PASSTHROUGH)
      ds.Close
```

  If you are using the data control, specify datacontrol.Database as the database variable as in this example:

```
      ' Enter the following two lines as one, single line:
      Set ds = Data1.Database.CreateDynaset("action statement",
         SQL_PASSTHROUGH)
```

STATUS
======

Microsoft has confirmed this to be a bug in the products listed above. This problem was fixed in the Compatibility Layer (COMLYR.EXE) update.

MORE INFORMATION
================

Steps to Reproduce Problem

The following program results in the incorrect "Invalid argument" error.

```
   Const DB_SQLPASSTHROUGH = &H40
   Dim db As Database
   Set db = OpenDatabase("", False, False, "ODBC")
   db.Execute "action statement", DB_SQLPASSTHROUGH
```

Additional reference words: 3.00 buglist3.00 fixlist3.00
KBCategory: kbinterop kbprg kbbuglist kbfixlist
KBSubcategory: APrgDataODBC

FIX: GPF with Long Formulas in Crystal Reports Custom Control
Article ID: Q108658

The information in this article applies to:

 - Professional Edition of Microsoft Visual Basic for Windows, version 3.0

SYMPTOMS
========

Loading the Formulas array property with long strings at run time can cause a general protection (GP) fault. The following problem applies to CRYSTAL.VBX, the Crystal Report custom control file.

CAUSE
=====

This is a memory management problem in the CRYSTAL.VBX control that ships with Visual Basic version 3.0.

RESOLUTION
==========

The newest version of CRYSTAL.VBX corrects this problem. You can download the latest CRYSTAL.VBX file by modem from the Crystal Services bulletin board system (BBS) at (604) 681-9516. In the Crystal Services BBS, download the VBVBX.ZIP file from the Files section.

STATUS
======

This bug is corrected by the latest version of CRYSTAL.VBX.

MORE INFORMATION
================

Steps to Reproduce Behavior
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Choose Add File from the File menu. Add the CRYSTAL.VBX file from your WINDOWS\SYSTEM directory. CRYSTAL.VBX is the Crystal custom control file.

3. Add a Crystal custom control to Form1.

4. Double-click the form to open the code window. Add the following code to the Form Load event:

```
   Sub Form_Load ()
      For i = 0 to 10
        Report1.Formulas(i) = Space$(200)
      Next i
   End Sub
```

5. Start the program, or press the F5 key. A GP fault may occur on some computers.

If you replace Space$(200) with Space$(110), the form loads but the GP fault may occur when you unload the form. If you change to Space$(100), the program may run without error. The exact behavior depends upon the current memory state of your Windows session.

REFERENCES

For a complete list of Crystal Reports support offerings, see the last three pages (PSS-1 to PSS-3) of the Microsoft Visual Basic Version 3.0, "Professional Features Book 2" in the Crystal Reports User's Manual section.

Additional reference words: 3.00 GPF buglist3.00 fixlist3.00
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: PrgCtrlsCus

FIX: Double-Click Still Maximizes/Restores If MaxButton=False
Article ID: Q110309

```
---------------------------------------------------------------------
The information in this article applies to:

 - Standard and Professional Editions of Microsoft Visual Basic for
   Windows, version 2.0
---------------------------------------------------------------------
```

SYMPTOMS
========

Setting the MaxButton property of a form to False removes the Maximize item in the Control-menu box and removes the maximize button in the upper right corner of the form. However, double-clicking the title-bar still maximizes the form or toggles back to the default size.

The Control-menu box is also known as the System-menu box in other products for Windows.

CAUSE
=====

By default, double-clicking the title bar has the same effect as choosing Maximize or Restore from the Control-menu box -- it acts as a toggle between the normal window size and the maximized window size. This behavior is by design in standard Microsoft Windows. Setting the MaxButton property of the form to False fails to suppress this behavior in Visual Basic version 2.0.

WORKAROUND
==========

To prevent a double-click on the title bar from causing Maximize or Restore, call Windows API functions as shown in the sample program in the More Information section below.

STATUS
======

Microsoft has confirmed this to be a problem in Visual Basic version 2.0 for Windows. This problem was corrected in version 3.0. In version 3.0, setting the form's MaxButton property to False correctly ignores double-clicks on the title bar.

MORE INFORMATION
================

The default Control-menu box in the upper left-hand corner of a Visual Basic form contains the following nine entries including separators:

```
   Restore
   Move
   Size
   Minimize
   Maximize
   -----------------------
   Close            Alt+F4
   -----------------------
   Switch to...   Ctrl+Esc
```

These are numbered 0 through 8 from the top down. You may remove any or all entries. Be sure to remove items in reverse sequence, from 8 to 0, or else the numbering will become confused.

NOTE: To remove the Control-menu box, set the ControlBox property to False. To remove the minimize button, set the MinButton property to False. To remove the maximize button, set the MaxButton property to False.

Steps to Work Around the Behavior

The following program removes the Maximize feature from a Visual Basic form. This code can be used in Visual Basic versions 2.0 and 3.0.

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add the following to the Form Load event code:

```
   Sub Form_Load ()

   Dim hSysMenu%, r%, j%, dw&, rr&
   Const MF_BYPOSITION = &H400

   ' Set the default size of the form:
   Form1.Height = Screen.Height + 45  ' Works on VGA.
   Form1.Width = Screen.Width + 60    ' Works on VGA.
   Form1.Left = -15                   ' Works on VGA.
   Form1.Top = -15                    ' Works on VGA.

   hSysMenu = GetSystemMenu(Form1.hWnd, 0)
   For j = 8 To 4 Step -1
      r = RemoveMenu(hSysMenu, j, MF_BYPOSITION)
   Next j
   For j = 2 To 1 Step -1
      r = RemoveMenu(hSysMenu, j, MF_BYPOSITION)
   Next j
   ' Leave Restore and Minimize in the Control-menu box.
   dw& = GetWindowLong(Form1.hWnd, -16)   'Window style
   dw& = dw& And &HFFFEFFFF            'Turn off Maximize button
   rr& = SetWindowLong(Form1.hWnd, -16, dw&)

   End Sub
```

3. Add a command button to the form. Double-click the command button and add the following code to the Command1 click event:

```
   Sub Command1_Click ()
      End
   End Sub
```

   This button lets you end the program because Close is removed from the Control-menu box.

4. Add the following Declare statements to the general declarations
   section:

```
   ' Enter each of the following Declare statements as one, single line:
   Declare Function RemoveMenu% Lib "User" (ByVal hMenu%, ByVal nPosition%,
      ByVal wFlags%)
   Declare Function GetSystemMenu% Lib "User" (ByVal hWnd%, ByVal revert%)
   Declare Function GetWindowLong Lib "User" (ByVal hWnd As Integer,
      ByVal nIndex As Integer) As Long
   Declare Function SetWindowLong Lib "User" (ByVal hWnd As Integer,
      ByVal nIndex As Integer, ByVal dwNewLong As Long) As Long
```

5. Start the program, or press the F5 key.

The form's Control-menu box shows Restore (greyed) and Minimize. Double-clicking the title-bar has no effect, as desired.

Clicking the Minimize arrow or choosing the Minimize menu item minimizes the form to an icon. A single-click on that icon does not open a control menu, unlike normal Visual Basic application icons. A double-click is required to restore the form to its full-screen state.

NOTE: In the above program, the following Form properties should be left with their design-time default: ControlBox = True, MaxButton = True, MinButton = True. The API functions take care of any necessary property changes.

REFERENCES
==========

 - "PC Magazine's Visual Basic Programmer's Guide to the Windows API" by Daniel Appleman (of Desaware), published by Ziff-Davis Press, pages 414 and 418. This reference describes most Windows API functions that can be used from within Visual Basic.

Additional reference words: buglist2.00 fixlist3.00 2.00
KBCategory: kbprg kbbuglist kbfixlist
KBSubcategory: PrgOther

FIX: Printer.Print Statements Unable to Print w/ Postscript
Article ID: Q113593

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows,
  versions 1.0 and 2.0

SYMPTOMS
========

Nothing prints on the paper when you try to print a string to the printer using the printer.print statement. Nothing prints when code processing reaches the printer.enddoc statement or the end of the program.

CAUSE
=====

The following two Postscript printer drivers were tested and found to have a problem with Microsoft Visual Basic versions 1.0 and 2.0 for Windows. The drivers with the problem are:

- PSCRIPT.DRV 318112 bytes dated 10-01-92

- PSCRIPT.DRV 313520 bytes dated 3-02-92

The problem is that nothing prints on the paper. The Printer.Print statement(s) may be ignored entirely when you are using either of the two drivers listed above.

WORKAROUND
==========

Add two statements prior to the Printer.Print statement(s) to correct the problem. You need to set the Printer.CurrentX and Printer.CurrentY properties to positive values. Then add the Printer.Print statement and you should see the desired string printed on the printer's page.

STATUS
======

Microsoft has confirmed this to be a bug in the products listed at the beginning of this article. This problem did not occur in Visual Basic version 3.0.

MORE INFORMATION
================

Note that you will need a PSCRIPT.DRV 318112 bytes 10-01-92 or PSCRIPT.DRV 313520 bytes dated 3-02-92 postscript printer driver to reproduce the problem outlined below.

Steps to Reproduce Problem
----

1. Start a new project in Visual Basic. Form1 is created by default.

2. Add a Command button (Command1) to Form1.

3. Add the following code to the Command1 click event procedure:

```
   Sub Command1_Click ()
      Printer.Print "This is a test of the Postscript printer driver"
      printer.enddoc
   End Sub
```

4. Run the program and click the Command1 button. You should see a blank piece of paper printed out if you have one of the specified drivers.

Additional Steps to Work Around the Problem
----

5. To work around the problem, add two lines of code to the Command1 click event procedure:

```
   Sub Command1_Click ()
      printer.CurrentX = 1
      printer.CurrentY = 1
      printer.Print "This is a test of the Postscript printer driver"
      printer.EndDoc
   End Sub
```

6. Run the program and click the Command1 button. Now you should see the line of text: 'This is a test of the Postscript printer driver' printed on the paper.

Additional reference words: buglist1.00 buglist2.00 fixlist3.00 1.00 2.00
buglist1.00 buglist2.00 fixlist3.00
KBCategory: kbpring kbprg kbfixlist kbbuglist
KBSubcategory: APrgPrint

UPD: GP Fault in KRNL286 When Run EXE on 286 or w/ NT on MIPs

Article ID: Q99251

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for
  Windows, version 3.0
---------------------------------------------------------------------

SYMPTOMS
========

You may encounter a general protection (GP) fault in KRNL286 at 0001:259F
or 0001:4FEC when you try to run a Visual Basic executable (.EXE) file in
Windows on a 286 computer or in Windows NT on a MIPs computer.

This problem will not occur when running a Visual Basic application from
the Visual Basic design environment on a 286 or MIPs computer.

RESOLUTION
==========

This problem has been fixed in a post-release version of VBRUN300.DLL,
which is available as part of self-extracting file named VBRUN300.EXE from
the Microsoft Software Library.

Download VBRUN300.EXE, a self-extracting file, from the Microsoft Software
Library (MSL) on the following services:

 - CompuServe
      GO MSL
      Search for VBRUN300.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN300.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN300.EXE

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft Visual Basic
programming system for Windows, version 3.0. To correct the problem, obtain
the post-release version of VBRUN300.DLL.

MORE INFORMATION
================

This bug occurs because of a problem with VBRUN300.DLL. The date,
time, size and version number of the VBRUN300.DLL file that leads to
this problem is as follows:
   Date: 04-APR-1993
   Time: 12:00 a.m.
   Size: 394384
   Version: 03.00.0537
```

The date, time, size and version number of the VBRUN300.DLL file that fixes this problem is as follows:

```
   Date: 12-MAY-1993
   Time: 12:00 a.m.
   Size: 398416
   Version: 03.00.0538
```

VBRUN100.DLL & VBRUN200.DLL Also Available in Self-Extracting Files

For your convenience, you can also obtain the .DLL files for Visual Basic versions 1.0 (VBRUN100.DLL in VBRUN100.EXE) and 2.0 (VBRUN200.DLL in VBRUN200.EXE). These files are not updates but are provided for your convenience.

Download VBRUN100.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for VBRUN100.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN100.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN100.EXE
```

Download VBRUN200.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for VBRUN200.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBRUN200.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \SOFTLIB\MSLFILES directory
      Get VBRUN200.EXE
```

Steps to Reproduce Problem in Visual Basic Version 3.0

----

1. Start 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. From the File menu, choose Make EXE File (ALT, F, k) and use the default name of PROJECT1.EXE.

3. Copy PROJECT1.EXE and VBRUN300.DLL to a 286 computer running Windows or a MIPs computer running Windows NT.

4. Run PROJECT1.EXE.

A GP fault occurs in KRNL286 at 0001:259F or 0001:4FEC.

Additional reference words: 3.00 GPF softlib update3.00 S14633 S14632 S14631
KBCategory: kbenv kbprg kbbuglist kbfixlist kbfile
KBSubcategory: EnvtRun

UPD: Oracle ODBC Setup and Connection Issues
Article ID: Q99706

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, version 3.0

SUMMARY
=======

The information given further below was taken from the latest version of the ORACLE.TXT file. A version of ORACLE.TXT was provided with Visual Basic version 3.0, but a later version (the one shown below) was provided with Microsoft Access version 1.1 for Windows. This updated version is provided in the More Information section below.

To install the earlier version of ORACLE.TXT on your computer, run Data Access Setup and install the Oracle ODBC driver. The ORACLE.TXT file will be installed in your WINDOWS\SYSTEM directory. Then you can update the file with the new information provided in the More Information section below.

The ORACLE.TXT file fails to mention that the SQL*NET drivers are not provided with Visual Basic. In order to use the information in this article, you must acquire the SQL*NET drivers from Oracle. You can contact Oracle at 1-800-345-DBMS.

MORE INFORMATION
================

SETTING UP THE ODBC ORACLE DRIVER FOR USE WITH THE SQL*NET FOR WINDOWS DLLs

This file discusses how to set up the ODBC ORACLE driver to run with your ORACLE Server software. To use the ODBC ORACLE driver with any large application, such as Microsoft Access, you must use the SQL*Net for Windows DLLs. Because the ODBC ORACLE driver is designed to use ORACLE Server version 6 and the SQL*Net for Windows DLLs are designed to use ORACLE Server version 7, you must be careful to configure your system correctly.

If you do not have the SQL*Net for Windows DLLs, or, after following the instructions in this file, you are still unable to connect to ORACLE Server with SQL*Net, you can contact Oracle Corp. at 1-800-345-DBMS.

If ORACLE Server Version 6 is Already Installed

To set up the ODBC ORACLE driver and the SQL*Net for Windows DLLs if you already have ORACLE Server version 6 on your system:

1. Make sure you have the correct versions of ORACLE products, including at least one SQL*Net protocol.

Product	Version
ORACLE Installer	3.0.8.3.7

Required Support Files	7.0.12.1.0
SQL*Net Named Pipes for Windows	1.1.1.3
SQL*Net SPX for Windows	1.1.1.5
SQL*Net TCP/IP for Windows	1.1.7.6

2. Test your current SQL*Net connection by using an ORACLE tool such as SQL*Plus for Windows.

3. Search for and delete all copies of ORA6WIN.DLL from your system. A new (backwards compatible) version of ORA6WIN.DLL will be installed with the ODBC ORACLE driver.

4. Run the ORACLE Installer program. When asked for your ORACLE installation directory, use the suggested default directory C:\ORAWIN.

5. Run the ORACLE Installer in the ORACLE group in the Program Manager:

   a) Install the files from the Required Support Files disk.

   b) Install the SQL*Net protocol you will be using. For more information, see the ORACLE documentation.

6. If the following line exists, remove it from your AUTOEXEC.BAT file:

   ```
   SET CONFIG=<oracle_configuration_file>
   ```

   Add the following line to your AUTOEXEC.BAT file:

   ```
   SET CONFIG_FILES=C:\WINDOWS\ORACLE.INI
   ```

   NOTE: If you are using the MS-DOS 6.0 operating system, add the following line to the end of your AUTOEXEC.BAT file:

   ```
   SET CONFIG=
   ```

   ORACLE Server first checks the CONFIG environment variable for the path of the ORACLE configuration file. If the CONFIG variable is not set, ORACLE Server checks the CONFIG_FILES variable. Because MS-DOS 6.0 can use the CONFIG environment variable during system startup, you must clear this variable before leaving your AUTOEXEC.BAT file. Otherwise, ORACLE Server will use its value as the path of the ORACLE configuration file.

7. Make sure your PATH variable includes the BIN subdirectories of your <oraclehome> directory and the \ORAWIN directory. For example, if your <oraclehome> directory is C:\ORACLE6, add the following line to your AUTOEXEC.BAT file:

   ```
   SET PATH=%PATH%;C:\ORACLE6\BIN;C:\ORAWIN\BIN
   ```

8. Paste the contents of your CONFIG.ORA file at the start of your ORACLE.INI file. For example, if your CONFIG.ORA file contains:

   ```
   LANGUAGE=American_America.US7ASCII
   ORACLE_HOME=C:\ORACLE6
   MACHINE_TYPE=J
   SQLPATH=C:\ORACLE6
   ```

```
  WIN_REMOTE_SESSIONS=3
  LOCAL=p:MyServer
```

and your ORACLE.INI file contains:

```
  [Oracle]
  ORACLE_HOME=C:\ORAWIN
  LANGUAGE=American_America.US7ASCII
  NLS_LANG=ENGLISH
  WIN_LOCAL_SESSIONS=1
  TCP_VENDOR=LANMAN
  TCP_SERVICES_FILE=C:\WINDOWS\SERVICES
```

then your modified ORACLE.INI file should contain:

```
  LANGUAGE=American_America.US7ASCII
  ORACLE_HOME=C:\ORACLE6
  MACHINE_TYPE=J
  SQLPATH=C:\ORACLE6
  WIN_REMOTE_SESSIONS=3
  LOCAL=p:MyServer

  [Oracle]
  ORACLE_HOME=C:\ORAWIN
  LANGUAGE=American_America.US7ASCII
  NLS_LANG=ENGLISH
  WIN_LOCAL_SESSIONS=1
  TCP_VENDOR=LANMAN
  TCP_SERVICES_FILE=C:\WINDOWS\SERVICES
```

NOTE: The ORACLE_HOME variable is set twice, once to point to the version 6 <oraclehome> directory and once to point to C:\ORAWIN.

9. If it is not already running, start Windows. Insert the ODBC Setup disk in drive A, choose Run from the Windows Program Manager (or File Manager) File menu, and then type "a:\setup.exe" in the Command Line box. For information about using the ODBC Setup program, see the online Help.

10. Run the ODBC Control Panel option and add a data source for your ORACLE server. For information about using the ODBC Control Panel option, see the online Help.

You should now be able to run the ODBC ORACLE driver. You should also be able to run ORACLE version 6 and version 7 tools and applications written for Windows. All of these can run over SQL*Net for Windows DLLs.

NOTE: Due to differences in memory use, this configuration may not allow you to run ORACLE MS-DOS-only tools or applications.

If ORACLE Server is Not Installed

To set up the ODBC ORACLE driver and the SQL*Net for Windows DLLs if you do not have any versions of ORACLE Server on your system:

1. Make sure that you have the correct versions of ORACLE products,

including at least one SQL*Net protocol.

Product	Version
ORACLE Installer	3.0.8.3.7
Required Support Files	7.0.12.1.0
SQL*Net Named Pipes for Windows	1.1.1.3
SQL*Net SPX for Windows	1.1.1.5
SQL*Net TCP/IP for Windows	1.1.7.6

2. Install the network software connecting your client workstation to the server and check that a connection can be made. For example, for the TCP/IP protocol, type "ping <servername>". This connection must work before you install the SQL*Net for Windows DLLs.

3. Run the ORACLE Installer program. When asked for your ORACLE installation directory, use the suggested default directory C:\ORAWIN.

4. Run the ORACLE Installer in the ORACLE group in the Program Manager:

   a) Install the files from the Required Support Files disk.

   b) Install the SQL*Net protocol you will be using. For more information, see the ORACLE documentation.

5. Add the following line to your AUTOEXEC.BAT file:

   ```
   SET CONFIG_FILES=C:\WINDOWS\ORACLE.INI
   ```

   NOTE: If you are using MS-DOS 6.0, add the following line to the end of your AUTOEXEC.BAT file:

   ```
   SET CONFIG=
   ```

   ORACLE Server first checks the CONFIG environment variable for the path of the ORACLE configuration file. If the CONFIG variable is not set, ORACLE Server checks the CONFIG_FILES variable. Because MS-DOS 6.0 can use the CONFIG environment variable during system startup, you must clear this variable before leaving your AUTOEXEC.BAT file. Otherwise, ORACLE Server will use its value as the path of the ORACLE configuration file.

6. Make sure your PATH variable includes the C:\ORAWIN\BIN directory. To do this, add the following line to your AUTOEXEC.BAT file:

   ```
   SET PATH=%PATH%;C:\ORAWIN\BIN
   ```

7. So that the ODBC ORACLE driver can use ORACLE version 7 error messages, copy the version 7 error messages to the directory where the ODBC ORACLE driver searches for error messages:

   ```
   COPY C:\ORAWIN\RDBMS70\*.MSB C:\ORAWIN\DBS
   ```

8. Search for and delete all copies of ORA6WIN.DLL from your system. A new (backwards compatible) version of ORA6WIN.DLL will be installed with the ODBC ORACLE driver.

9. If it is not already running, start Windows. Insert the ODBC Setup disk in drive A, choose Run from the Windows Program Manager (or File Manager) File menu, and then type "a:\setup.exe" in the Command Line box. For information about using the ODBC setup program, see the online Help.

10. Run the ODBC Control Panel option and add a data source for your ORACLE server. For information about using the ODBC Control Panel option, see the online Help.

You should now be able to run the ODBC ORACLE driver.

ORACLE Error Messages

The following section explains what to do when you encounter various error messages from ORACLE Server through the ODBC ORACLE driver.

ORA-xxxxx Message not found; product = RDBMS facility = ORA language = NULL

The ODBC ORACLE driver searches for error messages in the subdirectory that normally contains the ORACLE version 6 error messages. If you receive this error, it means that the ODBC ORACLE driver cannot find the error messages. To fix this:

1. Check that the CONFIG_FILES variable is set in your AUTOEXEC.BAT file and that it points to your ORACLE configuration file (ORACLE.INI). If you are using MS-DOS 6.0, check that the CONFIG environment variable is either not set or is cleared in the last line of your AUTOEXEC.BAT file.

2. Check that the ORACLE_HOME variable is set correctly in your C:\WINDOWS\ORACLE.INI file.

   If ORACLE Server version 6 was already installed on your system, ORACLE_HOME should be set twice. The first time, it should be set to your version 6 <oraclehome> directory, usually C:\ORACLE6. The second time, in the [Oracle] section of the file, it should be set to C:\ORAWIN.

   If ORACLE Server was not installed on your system, ORACLE_HOME should be set to C:\ORAWIN.

3. If you did not have any ORACLE software on your workstation, make sure that you copied all the .MSB files from C:\ORAWIN\RDBMS70 to C:\ORAWIN\DBS.

The ODBC ORACLE driver should now be able to print the ORACLE Server error message, enabling you to fix the problem that generated the error.

ORA-03121 No interface driver connected -- function not performed

The ODBC ORACLE driver cannot find ORA6WIN.DLL or one of the SQL*Net components. Check the following:

1. Without running the ODBC ORACLE driver, make sure the network connection is valid. For example, type "ping <servername>" for a TCP/IP connection.

2. Search for and delete old versions of ORA6WIN.DLL. The correct version of the ORA6WIN.DLL was installed by the ODBC ORACLE driver in the SYSTEM subdirectory of your Windows directory.

3. Check that the PATH variable contains the BIN subdirectory of the <oraclehome> directory (usually C:\ORACLE6\BIN or C:\ORAWIN\BIN).

4. Check that the CONFIG_FILES variable is set in your AUTOEXEC.BAT file and that it points to your ORACLE configuration file (ORACLE.INI). If you are using MS-DOS 6.0, check that the CONFIG environment variable is either not set or is cleared in the last line of your AUTOEXEC.BAT file.

5. Check that SQLTCP.DLL (for TCP/IP), SQLSPX.DLL (for Novell NetWare IPX/SPX), or SQLNMP.DLL (for Named Pipes) is in the ORACLE BIN directory specified in the PATH variable. (If not, SQL*Net was not installed correctly.)

6. Check that ORA7WIN.DLL and COREWIN.DLL are in the ORACLE BIN directory specified in the PATH variable. (If not, SQL*Net was not installed correctly.)

ORA-06120 NETTCP: network driver not loaded

This error can occur when ORA6WIN.DLL is loaded but cannot find another SQL*Net component, such as SQLTCP.DLL.

1. Check that the directories containing the SQL*Net components are in your PATH variable.

2. Check that the ORACLE_HOME variable is set correctly in your C:\WINDOWS\ORACLE.INI file.

   If ORACLE Server version 6 was already installed on your system, ORACLE_HOME should be set twice. The first time, it should be set to your version 6 <oraclehome> directory, usually C:\ORACLE6. The second time, in the [Oracle] section of the file, it should be set to C:\ORAWIN.

   If ORACLE Server was not installed on your system, ORACLE_HOME should be set to C:\ORAWIN.

3. Search for and delete old versions of ORA6WIN.DLL. The correct version of the ORA6WIN.DLL was installed by the ODBC ORACLE driver in the SYSTEM subdirectory of your Windows directory.

4. Check that you have followed all the instructions for the SQL*Net driver you are using. For example, for the SQL*Net for TCP/IP driver, make sure that all the TSRs, such as NMTSR and SOCKTSR, are loaded. (If not, SQL*Net was not installed correctly.)

ORA-0941: Error Translating Logical Name

This error may occur when you attempt to access an Oracle server via ODBC from Microsoft Access or Microsoft Visual Basic running on a client workstation that uses the SQL*Net TCP/IP protocol for Windows. When this error occurs, the CONFIG variable in your AUTOEXEC.BAT file that points to the ORACLE.INI file is invalid. To resolve this error, use the instructions in step number 6 of the "If ORACLE Server Version 6 is Already Installed" section above.

ODBC Error Messages

The following section explains what to do when you encounter various ODBC error messages.

IM003 Driver specified by data source could not be loaded

The ODBC Driver Manager is attempting to load the ODBC ORACLE driver (SQORA.DLL). SQORA.DLL loads ORA6WIN.DLL to connect to the ORACLE server. You can receive this message if it cannot find ORA6WIN.DLL or finds the wrong version of ORA6WIN.DLL.

1. Search for and delete old versions of ORA6WIN.DLL. The correct version of the ORA6WIN.DLL was installed by the ODBC ORACLE driver in the SYSTEM subdirectory of your Windows directory.

2. Make sure that ORA6WIN.DLL was installed when the ODBC ORACLE driver was installed.

Additional reference words: 3.00 ODBC Update3.00
KBCategory: kbinterop kbprg
KBSubcategory: APrgDataODBC

UPD: GENERIC Sample Not Provided with Visual Basic
Article ID: Q99888

The information in this article applies to:

- Professional Edition of Visual Basic for Windows, version 3.0

SYMPTOMS
========

Appendix E of the Control Development Guide in the "Microsoft Visual Basic Version 3.0 Professional Features Book 1" manual refers to a sample called GENERIC that it says is in the \SAMPLES\GENERIC subdirectory of Visual Basic. However, this sample was not provided with Visual Basic.

RESOLUTION
==========

You can get the GENERIC sample files by downloading a self-extracting file (GENERIC.EXE) from the Microsoft Software Library. After downloading the file, run it to obtain the GENERIC sample files.

Download GENERIC.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for GENERIC.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download GENERIC.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get GENERIC.EXE
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft Visual Basic programming system version 3.0 for Windows. This problem can be corrected by downloading the GENERIC sample files.

Additional reference words: 3.00 update3.00 softlib S14634
KBCategory: kbprg kbbuglist kbfixlist kbfile
KBSubcategory: PrgOther

UPD: New Setup Toolkit & Setup Wizard Available for VB ver 3.0

Article ID: Q100003

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SUMMARY
=======

Several bugs were fixed in the Visual Basic Setup Toolkit and Setup Wizard after Visual Basic version 3.0 was released. The latest release of each of these updated files (version 1.00.002 of SETUP.EXE, version 1.00.002 of the Setup1 project files, and version 1.00.551 of the Setup Wizard) can be found in the self-extracting file SETUPK.EXE. In addition, as of June 1994, SETUPK.EXE also includes WW1000.EXE, a self-extracting file that is also available separately in the Microsoft Software Library (MSL).

WW1000.EXE is a self extracting file that contains two files (README.TXT and VSHARE.386). VSHARE.386 currently ships as a component of Microsoft Windows for Workgroups versions 3.1 and 3.11, so you may already have it. The VSHARE.386 file is a driver that eliminates the need for SHARE.EXE when you run Windows version 3.1 in 386 enhanced mode. README.TXT is an Application Note that describes the installation and use of the VSHARE.386 driver. By installing and using VSHARE.386, you will prevent all known sharing violation errors when using the Setup Wizard to create installation disks. For more information, please see the "Problems and Limitations" section in this article.

NOTE: VSHARE.386 is exclusively a Windows utility. If you need a file sharing utility for MS-DOS, you still need to use SHARE.EXE.

How to Get SETUPK.EXE

Download SETUPK.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
  GO MSL
  Search for SETUPK.EXE
  Display results and download

- Microsoft Download Service (MSDL)
  Dial (206) 936-6735 to connect to MSDL
  Download SETUPK.EXE

- Internet (anonymous FTP)
  ftp ftp.microsoft.com
  Change to the \softlib\mslfiles directory
  Get SETUPK.EXE

MORE INFORMATION
================

The following sections outline the bugs that were fixed and give specific instructions on how to install the updated files.

Installation

After downloading SETUPK.EXE, run it in an empty directory to obtain the files it contains. Then copy the following files to the following locations on top of the existing files by the same name. (This assumes that you have installed Visual Basic in to the default directory C:\VB and Microsoft Windows in the C:\WINDOWS directory.)

```
SETUP.EXE         ->  C:\VB\SETUPKIT\KITFILES\SETUP.EXE
SETUP1.FRM        ->  C:\VB\SETUPKIT\SETUP1\SETUP1.FRM
SETUP1.FRX        ->  C:\VB\SETUPKIT\SETUP1\SETUP1.FRX
SETUP1.BAS        ->  C:\VB\SETUPKIT\SETUP1\SETUP1.BAS
SETUPWIZ.EXE      ->  C:\VB\SETUPKIT\KITFILES\SETUPWIZ.EXE
SETUPWIZ.INI      ->  C:\WINDOWS\SETUPWIZ.INI
                      and/or C:\VB\SETUPKIT\KITFILES\SETUPWIZ.INI
```

NOTE: Please update lines 2 through 5 of the SETUPWIZ.INI file to reflect the directory where you installed Visual Basic if it is different from the C:\VB directory.

The SETUPWIZ.INI file included here does not reflect the changes necessary to use the Access 2.0 Compatibility Layer files.

For more details on the Compatibility Layer, please see the following article(s) in the Microsoft Knowledge Base:

```
   ARTICLE-ID: Q113594
   TITLE     : Updated ACC2COMP.TXT for Jet 2.0/VB 3.0 Compatibility Layer
```

Distribution

You can distribute these files royalty free with any Visual Basic application that you create.

Setup Wizard Notes

To check the internal version number of your SETUPWIZ.EXE, open a copy of the file in an editor, and search for the string "ver:" This will show the version number. Visual Basic version 3.0 for Windows shipped with SETUPWIZ.EXE version 1.00.532.

Version	Bugs Fixed
1.00.533	When using "Save Template," you must enter a file name with extension. The extension is no longer required.
1.00.533	The "max" setting for the horizontal scrollbar on the Step Five screen is so large that the middle button of the scroll bar really can't be used. The max has been reset to a smaller value.
1.00.533	The standard command .PIF file may not have the EXECUTION = EXCLUSIVE on some computers. As a result, the DOS shells

	for compressing files may sit in the background. Now shell with parameter makes the task active with focus.
1.00.534	Minor tweaks to the user interface to widen the TEMPLATE buttons, added an accelerator to the R in Rebuild, and changed the accelerator in Exit to the 'x' key.
1.00.535	Start SetupWiz. Enter C:\. Click NEXT. This causes an untrapped error: "Path/file access error."
1.00.536	Removed line FILE10=OLE2UI.DLL under the [MSOLE2.VBX] section in SETUPWIZ.INI.
1.00.536	Try using Setup Wizard to create setup disks for the OLE2DEMO sample application after removing the OLE2UI entry from SETUPWIZ.INI. In step 2, select OLE. The Next button doesn't work. The Back button does work, and the Next button works if nothing is selected in step 2. The Finish button also works. This problem affected any OLE application.
1.00.537	Selecting more than 40 files with the Common Dialog during ADD FILES was not handled before. Now it is.
1.00.538	Cleaned up the MSOLE2.VBX and OLE Automation sections in SETUPWIZ.INI.
1.00.539	A PATH pointing to non-existing directories or drives resulted in a "User-defined error." Now Setup Wizard returns the correct error message and continues.
1.00.542	Fixed compression problems when running under Windows NT.
1.00.543	Fixed an invalid keyword in some common dialogs that asked where a file is located.
1.00.543	Fixed problem: if a template's .EXE file was deleted or moved.
1.00.544	SetupWizard incorrectly added VER.DL_ to the SETUP.LST file. Setup Wizard no longer adds this file to SETUP.LST.
1.00.545	SETUPWIZ.INI added two files to the CRYSTAL.VBX section.
1.00.546	Fixed problem where after adding files in step 5, you can get a "File not Found" or "Compress error."
1.00.546	Fixed problem where after deleting your project's EXE (such as MYAPP.EXE), you'd get an "Illegal function call in CreateVBSetup1."
1.00.547	Fixed problem where MYAPP.EXE in same dir as MAK file, and PATH= in MAK file points to different drive. Thus wrong file was added to list.
1.00.548	Fixed problem when a compressed file is larger than 1.2 meg. This fix also requires 4 changes in the SETUP1.MAK project.
1.00.548	Fixed problem when SETUP1.MAK has a .VBX or .DLL file.
1.00.548	Fixed problem where after adding multiple files, another 'point to a file' dialog came up asking for a support file location resulting in a path with no filename is listed in the file distribution box.
1.00.549	Fixed problems where SETUP1.FRM grows beyond an assumed size. SetupWizard is now not dependent on SETUP1.FRM's size.
1.00.550	Fix to the SETUP1.MAK files for concatenating split files back together.
1.00.551	Changed caption of Financial Functions checkbox to IFF & Financial Functions
1.00.551	Fixed problem where a project with many files would cause a Disk Full error on distribution disk 1.

SETUP.EXE Notes

To check the internal version number of your SETUP.EXE, open a copy of the file in an editor, and search for the string "FileVersion" to show the version number. Note, This version information was added only after version 1.00.002.

Version	Bug Fix/Feature	Comments
1.00.002	VER.DLL is truncated to zero bytes if it is not found or has an incorrect name on the distribution disk.	SETUP.EXE now checks to see if VER.DL_ exists on your distribution disk. If it is not found, the following error error is displayed and then SETUP.EXE terminates: "Error - File not found: A:\VER.DL_. This file is required by Setup."
1.00.002	SETUP.EXE does not run in Windows version 3.0.	When running SETUP.EXE in Microsoft Windows version 3.0, you will receive the error "This application requires a newer version of Windows." This error causes SETUP.EXE to terminate. This bug has been fixed so that SETUP.EXE will run successfully in Microsoft Windows version 3.0.
1.00.002	The Visual Basic version 3.0 THREED.VBX does not overwrite the Visual Basic version 2.0 THREED.VBX.	This problem occurs because the file type of THREED.VBX changed from "APP" to "DLL" between version 2.0 and 3.0. SETUP.EXE now ignores file type differences and will install any file where the source and destination names are the same when the source file is the same or a newer version.
1.00.002	This error message: "Could not open or read file: <filename>" was replaced with two separate error messages.	The error messages are now "Error - Could not open file: <filename>" and "Error - Could not read file: <filename>." Both errors cause SETUP.EXE to terminate.
1.00.002	New error message added: "Error - Insufficient disk space on drive <drive letter>:" This error causes SETUP.EXE to terminate.	The new error message replaces: "Error - Could not copy file: <source filename> -> <destination filename>" when there is insufficient disk space.

1.00.002	Version information was added to SETUP.EXE.	The version number 1.00.002 was added to SETUP.EXE. Previous versions of SETUP.EXE have no version number.
1.00.003	Running SETUP.EXE version 1.00.002 from a subdirectory causes "Error - Could not open file: <path name> SETUP.LST	This problem has been fixed so that you can run SETUP.EXE from a subdirectory. SETUP.EXE provided with Visual Basic version 3.0 does not have this problem.
1.00.003	VER.DL_ on distribution disk does not copy over VER.DLL in destination directory if the file date/ time stamp is the same for both files.	SETUP.EXE now copies VER.DL_ when the file/date time stamp of the destination VER.DLL file is the same.
1.00.004	A "Cannot copy file ..." error message occurs when attempting to copy a file referenced in SETUP.LST that has an older version number than the same file on the destination drive. This error causes SETUP.EXE to terminate.	SETUP.EXE no longer gives an error when the source file has an older version number than the destination file.
1.00.004	SETUP.EXE copies over the same or older version of VER.DLL if it is in use by another application such as File Manager. This can lead to a General Protection Fault (GP fault).	SETUP.EXE no longer attempts to copy VER.DLL if it is already in use. It assumes that it can use an older version of VER.DLL if it exists and is in use.

SETUP1 Project Files Notes

To check the internal version number of the Setup1 project files, check the general declarations section of SETUP1.FRM. (This version information was added only after version 1.00.002 of the Setup kit.)

Version	Bug	Comments
1.00.001	SETUP1.EXE fails to copy the Visual Basic version 3.0 THREED.VBX over the Visual Basic version 2.0 THREED.VBX.	This problem occurs because the file type of THREED.VBX c changed from APP to DLL between Visual Basic versions 2.0 and 3.0. The CopyFile function in SETUP1.BAS was modified so that any file will be copied regardless of its type as long as the source file is the same or newer version

		when compared to the destination file.
1.00.001	SETUP1.EXE fails when attempting to show a Program Manager group under Norton Desktop	A problem in CreateProgManItem when executing the ShowGroup DDE command causes SETUP1.EXE to fail under Norton Desktop. The syntax on the call to the ShowGroup DDE command was fixed to overcome this problem.
1.00.001	Unnecessary code included in Form_Load event procedure of SETUP1.FRM.	SETUP1.FRM contains code that has been disabled by making it into a comment. This code was useful in the Visual Basic version 1.0 of the Setup Toolkit. However, the features demonstrated by this code were removed from the Visual Basic version 2.0 and 3.0 Setup Toolkit. This code was removed.
1.00.001	Incorrect references to "Test Application"	Messages containing references to "Test Application" were changed to reference the actual name of the application.
1.00.002	Setup Wizard is not able to break large files (greater than 1.2 meg in size) across multiple disks	Changes to the Setup Wizard version 1.00.548 to fix this problem required changes to the CopyFile and ConcatSplitFiles routines in SETUP1.BAS.
1.00.002	A version number was added to a comment in the general declarations section of SETUP1.FRM.	Check the general declarations section of SETUP1.FRM to determine the current version number of SETUP1.

```
Changes Made to the CopyFile and ConcateSplitFiles Routines in SETUP1.BAS
--------------------------------------------------------------------------

Old SETUP1 Code:

   In Function CopyFile:

      If InFileVer$ <= OutFileVer$ Then

   In Sub ConcatSplitFiles:

      CopyLeftOver& = outfileLen& Mod 10
      CopyChunk# = (outfileLen& - CopyLeftOver&) / 10
      filevar$ = String$(CopyLeftOver&, 32)
      Get #fh2%, , filevar$
      Put #fh1%, , filevar$
      filevar$ = String$(CopyChunk#, 32)
      iFileMax% = 10
```

New SETUP1 Code:

In Function CopyFile:

```
      If InFileVer$ <= OutFileVer$ And SourcePath <> DestinationPath Then
```

In Sub ConcatSplitFiles:

```
      CopyLeftOver& = outfileLen& Mod 100
      CopyChunk# = (outfileLen& - CopyLeftOver&) / 100
      filevar$ = String$(CopyLeftOver&, 32)
      Get #fh2%, , filevar$
      Put #fh1%, , filevar$
      filevar$ = String$(CopyChunk#, 32)
      iFileMax% = 100
```

Problems and Limitations

- COMPRESS.EXE will take only a limited length command line. If SetupWizard is in a subdirectory that is nested too deep, COMPRESS will not work correctly. In this case, you will encounter a 'File does not exist' error when the file does exist. To work around this problem, move the SETUPKIT subdirectory up one or more directory levels until COMPRESS works.

- The Setup Wizard requires that your <appname>.EXE file completely reside on the first distribution disk. It will not split this file. If <appname>.EX_ is too large to fit on the first disk, an error will occur. To work around this problem, perform these steps:

  1. Start Visual Basic and create a temporary project file, <apptemp>.MAK. Give the project one codeless form and no custom controls. Save the project, create the <apptemp>.EXE file, and exit Visual Basic.

  2. Run the Setup Wizard specifying <apptemp>.MAK, and add your real project's .EXE file and all of the required supporting files during step 5 to create installation disks.

  3. Start Visual Basic and open the SETUP1A.MAK project file in the \VB\SETUPKIT\KITFILES directory and modify the Form_Load procedure in SETUP1A.FRM as follows:

     - Delete the following line:

```
          If Not CopyFile(SourcePath$, destPath$, "<apptemp>.EX_",
             "<apptemp>.EXE") Then GoTo ErrorSetup
```

     - Change the file name <apptemp>.EX_ in the following statement:

```
         If Not PromptForNextDisk(1, SourcePath$ + "<apptemp>.EX_")
            Then GoTo ErrorSetup
```

to the name of another file found of the first distribution disk, such as SETUP1.EX_:

```
         If Not PromptForNextDisk(1, SourcePath$ + "SETUP1.EX_")
            Then GoTo ErrorSetup
```

  - Create a new SETUP1.EXE

4. Shell out to MS-DOS to the \VB\SETUPKIT\SETUP1 directory and compress SETUP1.EXE using this command:

```
   \VB\SETUPKIT\KITFILES\COMPRESS SETUP1.EXE -R
```

5. Copy your new SETUP1.EX_ onto your first distribution disk.

- The Setup Wizard will not write to an installation disk that has damaged sectors, even if there is room for the project on it. This behavior is by design.

- The Setup Wizard will not split a file into more than nine pieces. If you need to ship a file that large, use another method to store the file.

- When doing step 1 of the Setup Wizard, if you select a valid .MAK file by using the common dialog and then reset it to C:\, you will cause an untrapped fatal error. This problem occurs only when you use the 'Select MAK file' option and then modify the result. This problem does not occur if the 'Select MAK file' option is not used.

- When doing step 5 of the Setup Wizard, if you add a large number of files, such as the entire WINDOWS directory, by using the common dialog box, the Setup Wizard:

  - Adds none of the file names.

  - Places a pause symbol (sideways =) in the list.

  - Does not generate an error message.

  This is caused by exceeding a buffer size limitation. You can delete the pause symbol without consequence.

- The Setup Wizard does not handle files that have nearly identical names correctly. If the names are identical except for the last letter (such as FORM1.FRM and FORM1.FRX), when the Setup Wizard compresses the second file, it does not check to determine if a file of that name (FORM1.FR_) already exists.

  To work around this problem, rename files that will cause this error before they are compressed. For example rename FORM1.FRX to FORM1X.FRX. To create the distribution disks, following these steps:

  1. Start Visual Basic, and open the SETUP1A.MAK project file in the \VB\SETUPKIT\KITFILES directory. Modify the Form_Load

procedure in SETUP1A.FRM as follows:

- Change the file name FORM1X.FRX in the following statement:

```
If Not CopyFile(SourcePath$, destPath$, "FORM1X.FR_",
   "FORM1X.FRX") Then GoTo ErrorSetup
```

  to the correct file name, FORM1.FRX:

```
If Not CopyFile(SourcePath$, destPath$, "FORM1X.FR_",
   "FORM1.FRX") Then GoTo ErrorSetup
```

- Create a new SETUP1.EXE

2. Shell out to MS-DOS in the \VB\SETUPKIT\SETUP1 directory, and compress SETUP1.EXE by using this command:

```
\VB\SETUPKIT\KITFILES\COMPRESS SETUP1.EXE -R
```

3. Copy your new SETUP1.EX_ onto your first distribution disk.

- Program manager Groups and Items may not install correctly on systems using an alternate desktop, such as Norton Desktop for Windows 2.2. For more information on this problem, please see the following article in the Microsoft Knowledge Base:

  ARTICLE-ID: Q108498
  TITLE   : PRB: DDE Error When Running Setup on Norton Desktop

- The error message 'Sharing Violation on drive C:' may be displayed during the compression stage (Step 6) when using the Setup Wizard. This is caused by the combination of the file sharing utility SHARE.EXE, the compression utility COMPRESS.EXE, and the Setup Wizard tool SETUPWIZ.EXE. The problem occurs when the compression utility tries to open the files SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, and/or CMDIALOG.VBX.

  To work around this problem, copy SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, and CMDIALOG.VBX from the \WINDOWS\SYSTEM directory to the directory where the SETUPWIZ.EXE file is located. Then SETUPWIZ.EXE and COMPRESS.EXE will not try to use the same files at the same time. Set the read-only attribute of all four files. This can be done via File Manager in Windows or by using the Attrib command from the DOS prompt.

  Users of Windows for Workgroups do not encounter this problem due to the fact that it uses VSHARE.386, an alternative file sharing utility to SHARE.EXE. Users of Windows 3.1 can also use this utility by installing VSHARE.386. Run the file WW1000.EXE, a self-extracting file included in SETUPK.EXE, to obtain the instructions (README.TXT) and the driver (VSHARE.386).

  For more information on this Bug, please see the following article in the Microsoft Knowledge Base:

  ARTICLE-ID: Q102478
  TITLE   : BUG: Setup Wizard Error: Sharing Violation Reading Drive C:

 - Filenames specified in SETUPWIZ.INI must be specified with all uppercase
   letters. This works around a problem with the Setup Wizard where files
   specified with the ":1" option are improperly installed into the same
   directory as the project .EXE file.

Additional reference words: Update3.00 softlib 3.00
KBCategory: kbtool kbfile kbprg kbcode
KBSubcategory: TlsSetWiz

UPD: New XBASE Driver Available That Fixes Several Problems
Article ID: Q100514

The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0

SUMMARY
=======

A new XBase IISAM driver XBS110.DLL version 1.10.0002 is available. This driver fixes several bugs documented below. It is the same driver that is provided with Microsoft Access version 1.10.

To obtain the new driver, download XBS110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
  GO MSL
  Search for XBS110.EXE
  Display results and download

- Microsoft Download Service (MSDL)
  Dial (206) 936-6735 to connect to MSDL
  Download XBS110.EXE

- Internet (anonymous FTP)
  ftp ftp.microsoft.com
  Change to the \SOFTLIB\MSLFILES directory
  Get XBS110.EXE

MORE INFORMATION
================

If you have Windows for Workgroups, you can use the following steps to get the version number of your XBase driver:

1. Start File Manager

2. Find the XBS110.DLL file, and select it. This file is usually located in the WINDOWS\SYSTEM directory.

3. From the File menu, choose Properties.

The item marked "Version:" is the XBase version number for XBS110.DLL.

Bugs Fixed by XBS110.DLL Version Number 1.10.0002

PROBLEM ID: 2186

Relates to DBase III

An update is allowed that violates unique index. Using the XB110.DLL driver that shipped with Visual Basic, it is possible to add multiple records that share the same unique index. The new version of the driver does not allow you to update the database with a record that contains the same unique index value as an existing record.

PROBLEM ID: 2390

Relates to FoxPro 2.5

A general protection (GP) fault occurs when updating the record immediately preceding a record locked by another user. The GP fault occurs in XBS110.DLL at 0002:11DA.

PROBLEM ID: 2418

Relates to DBase III

A unique index is corrupted after an update query. The symptom of this problem is that the first 239 items in the table are not found.

PROBLEM ID: 2432

Relates to DBase III, IV and Fox Pro 2.0, 2.5

SeekEQ on NULL returns first non-null record when there are no NULL records in the column.

PROBLEM ID: 2457

Relates to: FoxPro 2.5

Attempting to update a record results in a GP Fault in XBS110.DLL at 0013:144A when the IDX index type is used.

PROBLEM ID: 2487

Relates to FoxPro 2.5

A GP fault in XBS110.DLL occurs at 001A:05F6 when using INSERT INTO on the same table as the FROM clause uses -- that is, when copying records from a table into itself.

PROBLEM ID: 2511

Relates to FoxPro 2.0 and 2.5

A GP fault in XBS110.DLL occurs at 0002:11DA when inserting the 98th record in table that has one index.

Additional reference words: 3.00 update3.00 softlib S14644 GPF
KBCategory: kbenv kbprg kbbuglist kbfixlist kbfile
KBSubcategory: EnvtRun

UPD: Invalid file format Error When Run VB app's EXE File
Article ID: Q101261

The information in this article applies to:

- The Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

You may encounter the following error when running a Visual Basic executable (EXE) file:

```
   Invalid file format
```

Or you may encounter the following error when loading a Visual Basic project or form:

```
   Error loading '<form filename>'. A control could not be loaded due
   to a load error. Continue?
```

CAUSE
=====

This problem will occur when you have installed a new version of a custom control and the internal property list of the control has incorrectly changed in a way that breaks backward compatibility.

This problem is known to occur when you have installed the Visual Basic version 3.0 GRID.VBX file over an earlier version of the grid. Specifically, the problem will occur for an existing Visual Basic application, built using a previous version of the grid, that sets the HelpContextID property of the grid.

In the case where the problem occurs when you load a project into Visual Basic that contains a grid, the problem will only occur when the form file(s) containing the grid have been saved in binary format.

This problem is also known to occur when using Visual Basic version 2.0 and the CMDIALOG.VBX control. For more information on this problem, please see the following article in the Microsoft Knowledge Base:

```
ARTICLE-ID: Q100611
TITLE     : FIX: VB 2.0 Prof Demo Causes Error: Invalid File Format
```

WORKAROUND
==========

There are several ways that you can work around this problem:

If you are using a Visual Basic version 3.0 application and you encounter this problem, you can:

 - Acquire an updated copy of GRID.VBX from Microsoft (see instructions in the More Information section below).

- Replace the Visual Basic version 3.0 of GRID.VBX with an earlier version. A disadvantage of this strategy is that applications requiring the Visual Basic version 3.0 grid will not run.

If you are a developer of a Visual Basic version 3.0 application that uses the grid, you can:

- Acquire an updated copy of GRID.VBX from Microsoft (see instructions in the More Information section below). You will need to build your application using this grid.

- Rename GRID.VBX to a different name such as MSGRID3.VBX and rebuild the application using the renamed grid. A disadvantage of this strategy is that the grid will not be automatically updated when a new version of the grid (such as a version of the grid containing bug fixes) is released.

The following shows the date, time, size, and version number of the GRID.VBX file that leads to this problem:

```
  Date: 28-APR-1993
  Time: 12:00 a.m.
  Size: 44667
  Version: Not Marked
```

The following shows the date, time, size, and version number of the GRID.VBX file that fixes this problem:

```
  Date: 15-JUNE-1993
  Time: 5:26 p.m.
  Size: 45136
  Version: 03.00.0538
```

STATUS
======

Microsoft has confirmed this to be a bug in the Microsoft products listed at the beginning of this article. The problem is corrected by the updated version of GRID.VBX.

MORE INFORMATION
================

How to Obtain Updated Copy of GRID.VBX

To obtain the updated copy of GRID.VBX, download VBGRID.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
      GO MSL
      Search for VBGRID.EXE
      Display results and download

- Microsoft Download Service (MSDL)

```
      Dial (206) 936-6735 to connect to MSDL
      Download VBGRID.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get VBGRID.EXE
```

Steps to Reproduce Problem
----

1. Start 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 a Visual Basic version 1.0 or 2.0 version of GRID.VBX to Form1.

3. Put a grid control (Grid1) on Form1

4. Set the HelpContextID property of Grid1 to 1 (or some non-zero value).

5. From the File menu, choose Make EXE File (ALT, F, K) and create an EXE called PROJECT1.EXE.

6. Replace the older version of grid with the Visual Basic version 3.0 version of GRID.VBX, which has a date and time of 28-APR-1993 12:00 am.

7. Run the PROJECT1.EXE file created in step 5.

You should encounter an "Invalid file format" error. If you replace the Visual Basic version 3.0 grid with the version of the grid used in Step 2 and re-run PROJECT1.EXE, the program should run correctly.

Additional reference words: 3.00 softlib update3.00 S14643
KBCategory: kbprg kbfile kbfixlist kbbuglist
KBSubcategory: PrgCtrlsStd

UPD: New MSCOMM control available

Article ID: Q101944

The information in this article applies to:

- Microsoft Visual Basic for Windows, versions 2.0 and 3.0

SUMMARY
=======

A new version of the MSCOMM.VBX control is available. To obtain it, download MSCOMM.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for MSCOMM.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download MSCOMM.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get MSCOMM.EXE
```

This new control fixes a problem where the OnComm event will not fire when communicating with some 14.4K baud modems.

MORE INFORMATION
================

The date, time, size, and version number of the MSCOMM.VBX file that shipped with Visual Basic version 3.0 is:

```
   Date: 28-APR-1993
   Time: 12:00 a.m.
   Size: 34304
   Version: 2.0.9000.7
```

The date, time, size, and version number of the updated MSCOMM.VBX file is as follows:

```
   Date: 12-MAY-1993
   Time: 12:21 p.m.
   Size: 34816
   Version: 2.1.0.1
```

When using the MSCOMM.VBX provided with Visual Basic version 2.0 or 3.0 to communicate with a 14.4K baud modem, the OnComm event may not fire. The revised version of the MSCOMM.VBX control available on CompuServe fixes this problem by introducing a new Notification property. The problem relates to using Windows version 3.1 event driven communications. The new property fixes the problem by allowing you to use Windows version 3.0 polling techniques instead.

The Notification property is not available at design time, but you can get and set its property value at run time. It's default value of zero (0) tells the control to use Windows version 3.0 polling techniques. A value of 1 tells the control to use Windows version 3.1 event driven communications.

Microsoft recommends that you set the property value to 1 if you are using the MSCOMM control to communicate with a modem that has a baud rate lower than 14.4K baud.

One other change was made. The default property setting for the Interval property was changed from 1000 to 55.

Installation

To install the new control, copy the updated version to the WINDOWS\SYSTEM directory on your computer. Also check to make sure that no other copies of MSCOMM.VBX exist on your computer. If you find an older version of the MSCOMM.VBX file, delete it or rename it.

NOTE: When installing the updated MSCOMM.VBX, make sure no existing Visual Basic applications that used the original MSCOMM.VBX are broken. The updated MSCOMM.VBX (Version: 2.1.0.1) is not compatible with Visual Basic application .EXE files compiled with the original version of the MSCOMM.VBX (Version: 2.0.9000.7). These applications must be recompiled with the 2.1.0.1 version of the MSCOMM.VBX in order to work correctly with this updated .VBX file. The updated version of MSCOMM.VBX fixes a problem where the OnComm event will not fire when communicating with some 14.4 baud modems. If the specific problem fixed by the updated MSCOMM.VBX is not a concern and there are existing Visual Basic applications for which the source code is not available, use the original MSCOMM.VBX -- not the updated one.

Additional reference words: 2.00 3.00 update3.00 softlib S14642
KBCategory: kbprg kbfile
KBSubcategory: PrgCtrlsCus

UPD: New Access Engine MSAJT110.DLL Available

Article ID: Q102481

```
---------------------------------------------------------------------
The information in this article applies to:

- Standard and Professional Editions of Microsoft Visual Basic
  programming system for Windows, version 3.0
---------------------------------------------------------------------
```

SUMMARY
=======

A new Microsoft Access engine library MSAJT110.DLL version 1.10.0001 is available. To obtain it, download MSAJT110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for MSAJT110.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download MSAJT110.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get MSAJT110.EXE
```

MORE INFORMATION
================

This updated version of the MSAJT110.DLL is provided for compatibility. It is identical to the one that shipped with (and is required by) Microsoft Access version 1.1.

To get the version number of your current Access engine library, perform these steps:

1. Start File Manager.

2. Find the file MSAJT110.DLL, and select it. This file is usually located in the \WINDOWS\SYSTEM directory.

3. From the File menu, choose Properties.

The item marked "Version:" is the version number for MSAJT110.DLL.

Additional reference words: 3.00 1.10.0001 update3.00 softlib S14641
KBCategory: kbprg kbfile
KBSubcategory: APrgDataAcc

UPD: DOC: Data Access Guide Index -- A through Me

Article ID: Q103702

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SUMMARY
=======

Below is the A-Me index for the Data Access Guide in the Microsoft Visual Basic version 3.0 for Windows "Professional Features Book 2" manual. This index was not included in the manual. The index in the very back of the manual is for the Crystal Reports section of the manual only.

For the Mo through Z portion of the index, please see the following article in the Microsoft Knowledge Base:

ARTICLE-ID: Q103703
TITLE   : DOC: Data Access Guide Index -- Mo through Z

The entire index is also avaiable as one, single file. To obtain it, download DATAINDX.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
  GO MSL
  Search for DATAINDX.EXE
  Display results and download

- Microsoft Download Service (MSDL)
  Dial (206) 936-6735 to connect to MSDL
  Download DATAINDX.EXE

- Internet (anonymous FTP)
  ftp ftp.microsoft.com
  Change to the \softlib\mslfiles directory
  Get DATAINDX.EXE

MORE INFORMATION
================

Additional reference words: 3.00 update3.00 softlib S14640 docerr
KBCategory: kbprg kbfile kbdocerr
KBSubcategory: APrgDataAcc

UPD: DOC: Data Access Guide Index -- Mo through Z

Article ID: Q103703

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SUMMARY
=======

Below is the Mo-Z portion of the index for the Data Access Guide in the Microsoft Visual Basic version 3.0 for Windows "Professional Features Book 2" manual. This index was not included in the manual. The index in the very back of the manual is for the Crystal Reports section of the manual only.

For the A through Me portion of the index, please see the following article in the Microsoft Knowledge Base:

```
ARTICLE-ID: Q103702
TITLE     : DOC: Data Access Guide Index -- A through Me
```

Insert this index in front of the Crystal Reports section in "Professional Features Book 2." The entire index is also avaiable as one, single file. To obtain it, download DATAINDX.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for DATAINDX.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download DATAINDX.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get DATAINDX.EXE
```

MORE INFORMATION
================

Additional reference words: 3.00 update3.00 softlib S14640 docerr
KBCategory: kbprg kbfile kbdocerr
KBSubcategory: APrgDataAcc

UPD: List of Updated Files for Visual Basic

Article ID: Q104863

The information in this article applies to:

- Standard and Professional Edition of Microsoft Visual Basic for
  Windows, version 3.0

SUMMARY
=======

This article lists those files that were updated after Visual Basic version 3.0 shipped.

MORE INFORMATION
================

The following list shows the date of latest update, the name of the self-extracting .EXE file that contains the updated files, the names of the individual files that have been updated, and a brief description of what each file is used for in Visual Basic.

You can download the self-extracting files from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for <filename.EXE>
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download <filename.EXE>

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get <filename.EXE>
```

Date	File to Download	Updated Files	Description
3/7/94	BTR110.EXE	BTRV110.DLL	Btrieve IISAM Driver
3/7/94	DATAINDX.EXE	DATAINDX.DOC	Index for the "Data Access Guide"
3/7/94	GENRIC.EXE	\VB\CDK\GENERIC <15 files>	Sample custom control source code
3/7/94	VBGRID.EXE	GRID.VBX	Grid control
3/7/94	VBHC505.EXE	HC.EXE	Standard mode WinHelp Compiler
		HC.ERR	Error messages list for HC.EXE
		HCP.EXE	Protected mode WinHelp Compiler
		HCP.ERR	Error messages list for HCP.EXE
3/7/94	MSAJT110.EXE	MSAJT110.DLL	Microsoft Access Database Engine
3/8/94	MSCOMM.EXE	MSCOMM.VBX	Serial Communications control
3/7/94	ORA110.EXE	ORACLE.TXT	New updated ORACLE.TXT file
		SQORA.DLL	Oracle ODBC Driver file

		SQORASTP.DLL	Oracle ODBC Driver file
6/27/94	SETUPK.EXE	SETUP.EXE	Setup Toolkit
		SETUP1.FRM	Setup Toolkit
		SETUP1.FRX	Setup Toolkit
		SETUP1.BAS	Setup Toolkit
		SETUPWIZ.EXE	Application Setup Wizard
		SETUPWIZ.INI	Setup Wizard configuration file
		WW1000.EXE	Contains VSHARE.386 driver
3/7/94	VBRUN300.EXE	VBRUN300.DLL	Visual Basic Runtime Library
3/7/94	XBS110.EXE	XBS110.DLL	XBase IISAM Driver

There is an article in the Microsoft Knowledge Base that points to each of these files and that provides more detailed information about the update. To find these articles, query the Microsoft Knowledge Base using the file name and this word:

```
   update3.00
```

Additional files that were updated include the Compatibility Layer files OMLYR.EXE and BTR200.EXE.

Additional reference words: 3.00 update3.00 softlib
KBCategory: kbref kbfile
KBSubCategory: RefsDoc Setins

UPD: Updated BTRV110.DLL for Btrieve ISAM Driver shipped w/ VB

Article ID: Q112444

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0
- Microsoft Access, version 1.1

SUMMARY

An updated Btrieve driver (BTRV110.DLL) is available for use by registered owners of:

- Microsoft Access version 1.1
- Professional Edition of Visual Basic version 3.0

By downloading the new driver, you are indicating that you own one or both of these two products.

MORE INFORMATION

How to Get the New Driver

To get the updated driver (BTRV110.DLL), download BTR110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

- CompuServe
  GO MSL
  Search for BTR110.EXE
  Display results and download

- Microsoft Download Service (MSDL)
  Dial (206) 936-6735 to connect to MSDL
  Download BTR110.EXE

- Internet (anonymous FTP)
  ftp ftp.microsoft.com
  Change to the \softlib\mslfiles directory
  Get BTR110.EXE

What the Update Does for You

The updated BTRV110.DLL now uses text indexes, with exceptions as described in the section titled "Using Btrieve Indexes" of the file BTRIEVE.TXT, which ships with Microsoft Access version 1.1 and Visual Basic version 3.0.

How to Find Out If You Have the Updated Btrieve Driver Installed

The BTRV110.DLL file, which resides in the \WINDOWS\SYSTEM (local install) or the \ACCESS (network install) directory, should have a file size of 104560. The previous (old) driver had a file size of 104432.

If you have Microsoft Windows for Workgroups, you can use File Manager to determine if you have the correct version. Choose Properties from the File menu in File Manager to view information including the file version. The version of the updated BTRV110.DLL is 1.10.0011. The previous (old) driver file version was 1.10.0001.

How to Install the Update for Single-User Installations

Make a backup of your current BTRV110.DLL file by renaming the file or by copying it to another location. Then simply copy the updated BTRV110.DLL to your \WINDOWS\SYSTEM subdirectory.

How to Install the Update for Network Installations

Ask the network administrator to:

- Make a backup of the current BTRV110.DLL by renaming the file or by copying it to another location.
- Replace the old BTRV110.DLL file on the network in the \ACCESS directory with the updated BTRV110.DLL file.
- Make the new file available for those who installed Microsoft Access or Visual Basic on their local hard drives. In such a case, those users should also copy this file to their \WINDOWS\SYSTEM subdirectory.

```
===================================================================
************************** WARNING ********************************
Due to the nature of Microsoft Access' setup program, if setup is
run after updating the BTRV110.DLL file, the updated file will be
replaced with an older version of the file unless:

1. Setup /n is run and "No" is selected in answer to the question
   "Do you want to install Microsoft Access on your hard disk?" OR
2. The Custom Installation option is chosen, and then from the
   Microsoft Access Setup Options dialog the "xBASE, Paradox,
   Btrieve drivers" option is unchecked (or the Select button for
   this option is used to deselect the Btrieve option), OR
3. The Minimum Installation option is chosen.
===================================================================
```

Additional reference words: 3.00 softlib update3.00 S14639
KBCategory: kbtool kbfile kbinterop kb3rdparty
KBSubCategory: RefsProd

UPD: Windows 3.1 Help Compiler & Difficulty w/ Word 6.0 RTF

Article ID: Q112445

The information in this article applies to:

 - Microsoft Word for Windows, versions 6.0 and 6.0a
 - Microsoft Windows operating system version 3.1
 - Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS
========

The Windows version 3.1 Help Compiler version 3.10.504 (and earlier versions) will not function correctly with Microsoft Word version 6.0 rich-text format (RTF) output. You may receive a general protection (GP) fault during help file compilation or the help file may be compiled using only the Windows system font.

CAUSE
=====

This is caused by the way previous versions of the Help Compiler interpret new RTF controls relating to the font header information that Word version 6.0 generates when saving as RTF.

WORKAROUND
==========

To obtain the latest release of these updated Compiler (HC.EXE and HCP.EXE version 3.10.505), download VBHC505.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for VBHC505.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download VBHC505.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get VBHC505.EXE
```

MORE INFORMATION
================

Files in VBHC505.EXE

HC.EXE -- Standard mode Help Compiler
HCP.EXE -- Protected mode Help Compiler (Uses Expanded Memory)

HC.ERR -- Error file used by HC.EXE
HCP.ERR -- Error file used by HCP.EXE

Additional Problems Corrected

- More entries (approximately 8,000) are allowd in the .HPJ file in the map section and alias sections.

- A larger number of bitmap definitions (approximately 2,970) are allowed in an RTF input file.

- The internal compiler limit was increased from 32K to 64K.

Workaround for Problem Displaying Apostrophe in Help File

As the default, Microsoft Word versions 6.0 and 6.0a use Smart Quotes. However, the current version of the Help compiler ignores Smart Quotes in an .RTF file. If you have a problem displaying an apostrophe in a Help file, this is probably the cause.

To work around the problem in Microsoft Word:

1. From the Tools menu, choose AutoCorrect. Clear the first check box, "Change Straight Quotes to Smart Quotes."

2. From the Tools menu, choose Options. Click the AutoFormat tab in the upper-right corner. In the Replace frame, clear the "Straight Quotes with Smart Quotes" check box.

Additional reference words: textconv conversion converted converts transfer transfers translation translate problem 6.00 gpfault gpf hang hung crash crashed locks locked frozen freezes crashing quit quits stopped 3.00 update3.00 softlib S14638 6.0a hc505 hc505.exe
KBCategory: kbtool kbref kbfile
KBSubCategory: RefsProd

UPD: SQORA.DLL Does Not Allow Lengthy SQL Statements

Article ID: Q112446

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0
- Microsoft Access, version 1.1

SYMPTOMS
========

If the table and field names are long or the query is complex, executing a query or updating a record in an Oracle table results in the following error message:

```
   Statement was longer then allowable maximum 2000+ chars
```

CAUSE
=====

This occurs because of a problem with SQORA.DLL, the Oracle ODBC driver.

RESOLUTION
==========

Obtain and install the updated driver (instructions are provided in the More Information section below), or use queries to do updates rather than updating records with the Oracle table in Datasheet view. The query should yield only the columns to be updated. For complex queries, reduce both the number of tables or joins in the query and the number of fields used or shown in the query. This reduces the lengths of SQL statements.

STATUS
======

Microsoft has confirmed this to be a problem in the Oracle ODBC driver shipped with Microsoft Access version 1.1 and the Professional Edition of Microsoft Visual Basic version 3.0. An updated driver that corrects this specific problem is available for owners of Microsoft Access version 1.1 or the Professional Edition of Microsoft Visual Basic version 3.0.

MORE INFORMATION
================

How to Obtain the Updated Driver

The updated Oracle ODBC driver (SQORA.DLL) is available for use by registered owners of:

- Microsoft Access version 1.1
- Professional Edition of Visual Basic version 3.0

By downloading the new driver, you are indicating that you own one or both of these two products. To obtain the updated driver, download and then run

ORA110.EXE, a self-extracting file.

Download ORA110.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for ORA110.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download ORA110.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get ORA110.EXE
```

Contents of ORA110.EXE

README.TXT - a copy of this article
ORACLE.TXT
SQORA.DL_
SQORASTP.DL_
ODBC.INF
SETUP.EXE

NOTE: The SETUP.EXE file is called by the ODBC control panel facility and will not run as a stand-alone file.

How to Install the Updated Driver
----

1. Start Windows if it is not running.

    - If you are running Windows 3.1, open Control Panel.
    - If you are running Windows 3.0 or NT, select the ODBC program group.

2. Double-click the ODBC icon.

3. From the Data Sources dialog, select the Drivers... button.

4. From the Drivers dialog, select Add...

5. Enter the drive letter and directory from which you are installing.

6. Select Oracle from the list of available drivers, and choose OK. ODBC setup will install the driver at this point. If an ODBC Oracle driver of the same version number or higher exists on the hard disk, ODBC setup will ask if you want to replace it. In most cases, you will want to stay with the most recent version.

7. Choose close, and you are finished.

What ODBC Setup Installed

The ODBC installation installed a new SQORA.DLL, a new SQORASTP.DLL, and a new ORACLE.TXT to your Window's system directory.

```
                        Old                     New
                ----------------------------------------------
SQORA.DLL       Version:  1.00.2816     Version:  1.00.3112
                   Size:  143,600 bytes    Size:  144,096 bytes
                   Date:  4/16/93          Date:  7/12/93

SQORASTP.DLL    Version:  1.00.2403     Version:  1.00.3106
                   Size:  9,328 bytes      Size:  9,632 bytes
                   Date:  5/7/93           Date:  7/6/93
```

Oracle drivers are manufactured by Oracle Corporation and Btrieve drivers by Novell, Inc. These two vendors are independent of Microsoft; we make no warranty, implied or otherwise, regarding these products' performance or reliability.

Additional reference words: 1.10 3.00 update3.00 softlib S14637
KBCategory: kbinterop kbfile kbbuglist kbtool
KBSubCategory: RefsProd

UPD: Project 4.0 Files for ODK Encore Example Available in MSL

Article ID: Q117262

The information in this article applies to:

- Microsoft Office Developer's Kit version 1.0
- Professional Edition of Microsoft Visual Basic for Windows, version 3.0

The Encore example program, which is included on the Microsoft Office Developer's Kit (ODK) version 1.0 CD that ships with the Professional Edition of Visual Basic version 3.0, was designed to work with two Microsoft Project version 4.0 files (ENCORE!.MPP and GLOBAL.MPT). Because the ODK CD was released before Microsoft Project version 4.0 was available, the ENCORE!.MPP and GLOBAL.MPT files were not included on the CD.

To get the two files, download ODKPR4.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for ODKPR4.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download ODKPR4.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get ODKPR4.EXE
```

After downloading ODKPR4.EXE, run it in an empty directory to extract these files:

```
   ENCORE!.MPP
   GLOBAL.MPT
   READ_ME.TXT
```

To use the Microsoft Project version 4.0 files with the Encore example, copy ENCORE!.MPP and GLOBAL.MPT to the Encore example code directory. The Encore example code in ENCORVBA.XLS will look for the ENCORE!.MPP and GLOBAL.MPT files in its directory and enable the Microsoft Project version 4.0 functionality if it finds them.

Additional reference words: softlib 1.00 3.00 softlib
KBCategory: kbole kbprg kbfile
KBSubcategory: IAPOLE

UPD: OLE DBCS Enhancement Release Files Available
Article ID: Q119024

The information in this article applies to:

 - Professional Edition of Microsoft Visual Basic for Windows,
   version 3.0

SUMMARY
=======

A new set of OLE 2.01 dynamic-link libraries (DLLs) is available for download on CompuServe.

MORE INFORMATION
================

The OLE 2.01 Double-Byte Character Set (DBCS) Enhancement consists of (a) an update to the OLE 2.0 libraries that adds double-byte character support and (b) a new Microsoft License Agreement. These new OLE libraries supersede the files of the same name that were released on the OLE 2.01 SDK CD-ROM in the \SHIPWITH directory, and they must be redistributed with your OLE 2.0 application.

To obtain the updated files, download OLEDLL.EXE, a self-extracting file, from the following service:

```
 - CompuServe
      GO WINOBJ
      Search for OLEDLL.EXE
      Display results and download
```

To update your OLE libraries, you simply need to copy all the files in this self-extracting archive to your \WINDOWS\SYSTEM subdirectory.

Included in the archive is the Write file UPDATE.WRI, which contains additional information regarding licensing, use, and other modifications for using the OLE 2.01 SDK.

Additional reference words: 3.00 OLE2
KBCategory: kbole kbprg kbcode
KBSubCategory: IAPOLE

UPD: New Btrieve Driver BTRV200.DLL Available
Article ID: Q119739

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows,
  version 3.0

SUMMARY
=======

A new Btrieve driver, BTRV200.DLL version 2.0.0.15, is available. This driver is for use by registered owners of Microsoft Access version 2.0 and Visual Basic version 3.0 for Windows (Professional Edition only). By downloading this driver, you are indicating that you are a registered owner of one of these products.

To obtain the Btrieve driver, download BTR200.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for BTR200.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download BTR200.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the \softlib\mslfiles directory
      Get BTR200.EXE
```

MORE INFORMATION
================

This updated version of the BTRV200.DLL file is provided to fix a bug that causes the following error message when working with an attached Btrieve table:

```
   Error 3275: "Unexpected error from external database driver ()
```

The same bug can cause a query based on a Microsoft Access version 2.0 database with an attached Btrieve table to not return any records. The query works correctly on Microsoft Access version 1.x databases.

The problems stated above are caused by a bug in the BTRV200.DLL Btrieve ISAM driver that is installed by the Microsoft Jet 2.0/Visual Basic 3.0 Compatibility Layer.

If you are using Microsoft Access version 2.0, please refer to the following article in the Microsoft Knowledge Base, which has instructions for installing the updated BTRV200.DLL driver on both network and standalone installations of Microsoft Access version 2.0:

```
   ARTICLE-ID: Q116383
   TITLE:      Unexpected Error from External Database Driver [7]

Additional reference words: 3.00 update3.00 softlib
KBCategory: kbprg kbfile
KBSubCategory: APrgIISAM
```

UPD: Microsoft Access 2.0 Owners Can Get Updated Jet 2.5

Article ID: Q126387

The information in this article applies to:

- Microsoft Access, version 2.0
- Standard and Professional Editions of Microsoft Visual Basic for Windows, version 3.0

NOTE: You must own Microsoft Access version 2.0 for the files discussed in this article to be of use to you.

SUMMARY
=======

The Microsoft Access version 2.0 Service Pack contains files that replace existing files in your current Microsoft Access version 2.0 installation. The Service Pack includes the latest versions of:

- The Microsoft Jet database engine (version 2.5).

- The OLE dynamic-link libraries (version 2.02).

- The Btrieve, Paradox, and xBASE installable ISAM drivers.

You can obtain the Service Pack by downloading the files from one of the services listed below. Or, you can obtain the Service Pack by calling the Microsoft FastTips system and having the disks mailed to you.

MORE INFORMATION
================

Download Services

To obtain the Service Pack by downloading the files:

Download ACCSVC.EXE, a self-extracting file, from the Microsoft Software Library (MSL) on the following services:

```
 - CompuServe
      GO MSL
      Search for ACCSVC.EXE
      Display results and download

 - Microsoft Download Service (MSDL)
      Dial (206) 936-6735 to connect to MSDL
      Download ACCSVC.EXE

 - Internet (anonymous FTP)
      ftp ftp.microsoft.com
      Change to the SOFTLIB\MSLFILES directory
      Get ACCSVC.EXE
```

Microsoft FastTips

FastTips is an interactive telephone system that you can call 24 hours a day, 7 days a week for information on many Microsoft products. You can order the Service Pack to be sent by mail for no charge by dialing the FastTips line at 1-800-936-4100 and following these steps:

1. Press 4 for Microsoft Access.

2. Press 1 for version 2.x.

3. Press 1 for Express Order Service.

4. When you are prompted, press 2 for a mail delivery method.

5. At the tone, enter the number 1124 for the Item ID.

6. Press # to verify the order is correct.

7. When you are prompted, leave your name and mailing address in the voice mailbox.

8. When you are prompted, press * to finish your order.

REFERENCES
==========

For additional information on the Microsoft Access Service Pack, please see the following articles in the Microsoft Knowledge Base:

```
ARTICLE-ID: Q122927
TITLE     : WX1124: Microsoft Access Version 2.0 Service Pack

ARTICLE-ID: Q123708
TITLE     : INF: Directory Contents of Service Pack File ACCSVC.EXE

ARTICLE-ID: Q123823
TITLE     : INF: MS Access Version 2.0 Service Pack Questions and Answers

ARTICLE-ID: Q123590
TITLE     : INF: Fixes to xBASE ISAM in MS Access Service Pack

ARTICLE-ID: Q123592
TITLE     : INF: Fixes to Paradox ISAM in MS Access Service Pack

ARTICLE-ID: Q123594
TITLE     : INF: Fixes to Btrieve ISAM in MS Access Service Pack
```

Additional reference words: 3.00 accsvc update3.00
KBCategory: kbref kbfile
KBSubcategory: RefsProd