Microsoft Visual Basic for Windows: Bugs, Fixes, & Updates
Prepared 12/06/93

Unfixed Bugs

Fixed Bugs

Updates Available

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

Unfixed Bugs

- BUG: TABs Paste Incorrectly as | to VB.EXE's Immediate Window
- BUG: Scroll Box Flashing Not Updated If Bar Resized w/ Focus
- BUG: [Character May Sort Incorrectly in List or Combo Box
- BUG: Can Click in Code Window Without Activating it in VB.EXE
- BUG: Pressing ESC or CTRL+BREAK Makes Mouse Pointer Disappear
- BUG: No Beep When Click Form and the Menu Design Window Is Up
- BUG: Incorrectly Accessing System Menu of Hidden Form
- BUG: ExtFloodFill Won't Fill Over QBColors If AutoRedraw=True
- BUG: Duplicate Procedure Name Alters Original Capitalization
- BUG: No Option Button Active (Dotted) in Frame
- BUG: Dir List Box Does Not Give Error 68 Device Unavailable
- BUG: FormName Not in Correct Order After Out of Memory Error
- BUG: LinkTimeOut of -1 Waits Only 6553.5 Secs Before Time Out
- BUG: DateSerial Does Not Give Error for Invalid Month or Day
- BUG: Incorrect Focus Shift for Disabled Control in Break Mode
- BUG: Extra Click Event if Double-Click When Mouse Button Down
- BUG: CTRL+LEFT/RIGHT ARROW Behaves Differently When Edit/Type
- BUG: ToolBox Picture Control Bitmap Too Small on EGA
- BUG: Using Nonstandard Icons Can Cause UAE/GP Fault/Hang
- BUG: Right Mouse Button Causes Remote Control Menus
- BUG: Multiline Text Box Contents Not Gray When Enabled=False
- BUG: Visual Basic Code Window Hides Split View if Resized
- BUG: Invalid outside Sub Error When Copy or Paste to General
- BUG: Resetting ListIndex Property Generates Click Event
- BUG: Some Property Values May Be Incorrect in Maximized Form
- BUG: Option Button w/ Focus Selected When Click Form Caption
- BUG: Click Event May Fail to Occur in Cascading Menu
- BUG: TAB Character Can Incorrectly Cause KeyUp/KeyDown Events
- BUG: MDI Child CTRL+INSERT in Properties List Causes UAE/GPF
- BUG: No Resources Causes Failed to Open Graphics Server Error
- BUG: Gauge Custom Control: No Error for Illegal NeedleWidth
- BUG: MDI Child Left/Top Property Wrong in Properties Bar
- BUG: MDI Child Control: Large Height/Width Value Not Accepted
- BUG: Grid Custom Control: Scroll Bars Displayed Unnecessarily
- BUG: Gauge Custom Control: Valid NeedleWidth Range 1 to 32767
- BUG: 3-D Panel Control Doesn't Resize to Key Status Control
- BUG: Vertical Linear Gauge Loses Upper Border's Bottom Pixels
- BUG: InnerBottom/InnerRight Defines Gauge Fill Area Badly
- BUG: Graph: ExtraData May Not Say: Invalid Property Value
- BUG: 3D Command Button Shows Outline when Outline = False
- BUG: Scroll Control: UAE/GPF If Drag Method in GotFocus Event
- BUG: Grid: No Error Changing FixedAlignment on Non-Fixed Col

- BUG: Graph: AutoInc Increments ThisPoint Instead of ThisSet
- BUG: Animated Button: 8 Pt. Roman/Mdrn Fonts Don't Underline
- BUG: Graph Axis Titles Don't Switch on Horizontal Bar Graphs
- BUG: Menu Can Cover Top of MDI Child Control If BorderStyle=0
- BUG: VB Graph Custom Control: SeeThru Paints Incorrectly

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

- BUG: Some Controls Not Printed with PrintForm in Windows 3.1
- BUG: THREED.VBX: Command/Group Push Buttons Show Invalid File
- BUG: Common Dialog Custom Controls Don't Display Printer Fonts
- BUG: MDI Child Control Skips Index with Control Array
- BUG: Generic / Text Only Printer Driver Prints 66 Lines
- BUG: Illegal function call / Division By Zero Errors
- BUG: Stack Fault When Move Sets Tiny Width in 2-Item Combo Box
- BUG: GPF/UAE If Multi-Select Controls w/ No Common Properties
- BUG: Type Mismatch Error If Use VAL Function on Big Hex Value
- BUG: Stack Fault May Occur If Trapping Divide By Zero
- BUG: GPF When Close Form That Contains a Single MCI Control

BUG: Neg ScaleHeight Resizes Control When Form Saved as ASCII

- BUG: Stack Fault When Move Makes Combo Box Width Too Small
- BUG: Unable to Edit LinkNotify Event If Control Has Long Name
- BUG: ODBC Getchunk Method on Non-Memo Field Causes GPF/UAE
- BUG: OLE DataText Prop Doesn't Free Memory When Object Closed
- BUG: Changing Default Printer Doesn't Effect Printer.Fonts
- BUG: Wrong Menu Click Event After Hiding Menu
- BUG: MaskedEdit MaxLength Reset to 64 When Mask=""
- BUG: Overflow Error When CurrentX Or CurrentY Greater Than 32K
- BUG: VB Pro Setup Fails to Correctly Associate .HLP Files
- BUG: Out of Memory Error on Show Next from Debug Menu
- BUG: 3D Button Loses 256-Color Palette When Load 2nd Bitmap

BUG: Grid Control Repaints When Another Form Is Made Active

- BUG: Unload in 3D GroupPush Button Causes GP Fault
- BUG: Referencing Data Object Gives Error: Object not an Array
- BUG: GPF in Some Video Drivers When Load RLE Bitmaps > 20K
- BUG: Font3D Property Set Incorrectly in THREED.VBX Controls
- BUG: Data Access Setup Can Give Incorrect Error Message
- BUG: Ref to NPV / IRR / MIRR Gives Undefined Functions Error
- BUG: Incorrect Result When Multiple Aggregate Functions in SQL
- BUG: Incorrect Behavior in MaskedEdit BorderStyle Property
- BUG: Problems Printing Projects to HPLJ4
- BUG: ALT+MINUS SIGN Does Not Work with Maximized MDI Forms
- BUG: GP Fault When Opening Menu Design Window in VB.EXE

BUG: VB Dynasets Incorrectly Bypass Defaults on SQL Server

- BUG: Bad Result If Multiple Aggregate Functions in SQL Stmt
- BUG: Out of Memory w/ Var Named ClientLeft/Top/Width/Height
- BUG: Setup Wizard Error: Sharing Violation Reading Drive C:

- BUG: Domain Functions Available Only Within SQL Statement
- BUG: Can't Load Custom Control DLL: PICCLIP.VBX in Windows 3.0
- BUG: Out of Memory w/ MSOLE2.VBX When SHARE.EXE Not Loaded
- BUG: Invalid Argument Err on Execute Method w/ SQL Passthrough
- BUG: GPF in VB.EXE at 0038:3B6F w/ Compile-Time Error & Set
- BUG: Error 13 (Type Mismatch) & Error 3061 w/ SQL Queries
- BUG: Overflow in VB version 3.0 ICONWRKS Sample Program
- BUG: VB Printer.Width/Height Values Incorrect for Plotter
- BUG: VB Setup Files Modified or Corrupted, Using \WINDOWS Path
- BUG: Name Not Found in This Collection When Deleting Member
- BUG: Incorrect VB Error When Delete Index on Open Table

Fixed Bugs

Updates Available

Visual Basic for Windows: Bugs, Fixes, & Updates

Unfixed Bugs

Fixed Bugs

- FIX: VB Debug.Print in MouseMove Event Causes MouseMove Event
- FIX: Overflow in VB Drawing Circle Segment w/ Radius of Zero
- FIX: UAE When Place More than 64K in VB List Box or Combo Box
- FIX: Pull-Down on Drive Box Disabled When Change Width of Box
- FIX: UAE/GPF Changing MS-DOS Win Display If VB at Breakpoint
- FIX: Overflow Error If Print Long String to Form or Printer
- FIX: Control Overlaid by 2nd Control Won't Refresh If Moved
- FIX: Open Project Dialog Misbehaves If Project Dir Deleted
- FIX: Text Not Highlighted When Copy Immediate Win to Clipboard
- FIX: Undocumented Separator Property of a VB Menu Item
- FIX: Can't Have Menu with No Caption Bar/Buttons/Control Box
- FIX: ControlBox Property False Disables Focus w/ Keys in Menus
- FIX: StretchBlt() Gives UAE/GPF with 256-Color Video Drivers
- FIX: Visual Basic List Box Won't Open if Resized at Run Time
- FIX: Text Too Narrow with Italic Fonts in Visual Basic Labels
- FIX: SendKeys Causes Erratic Mouse Behavior on IBM PS/2
- FIX: DDE from Excel to VB Ver 1.0 Uses Up Windows GDI Heap
- FIX: File Not Loaded If No Extension in Load Picture Dialog
- FIX: Panel Custom Control Caption Not Dimmed When Disabled
- FIX: Graph Custom Control Incompatible w/ HP II Series Printer
- FIX: Animated Button Custom Control: Caption May Be Truncated
- FIX: Graph Control's Negative Values Plot As Positive
- FIX: Gauge: Incomplete Paint with Max-Min Difference > 100
- FIX: Grid: Changing Font Properties Resets ColWidth, RowHeight
- FIX: VB Graph Custom Control: BottomTitle Text May Disappear
- FIX: Outline Transparent in 3D Button When Outline=False
- FIX: Graph Custom Control: LabelText May Overlap
- FIX: Graph Custom Control Legends May Print Incorrectly
- FIX: Grid Cell Border May Not Display with Some BackColors
- FIX: Omitting Year for DateValue May Give Unexpected Results
- FIX: Toolkit 3-D Option & Check Controls Don't Repaint in 3.1
- FIX: Toolkit Setup Routine Causes Out of String Space Error
- FIX: Grid Custom Control RemoveItem Does Not Update RowHeight
- FIX: GP Fault or UAE When Unload Form in DragOver Event
- FIX: UAE/GPF Occurs If EXE Uses Variable Length String in Type
- FIX: UAE/GPF When Use Static Array in Event Procedure After F5
- FIX: UAE/GPF When VB Control Name Identical to Property Name
- FIX: UAE/GPF When Square Brackets '[]' Around MSGBOX Function
- FIX: GPF/UAE When Converting String > 32K to Double Precision
- FIX: VB Painting Problem Occurs When Low on System Resources
- FIX: Result Differs When Comparing Single w/ Double Precision

- FIX: GPF/UAE When Closing DDE Application from the Task List
- FIX: GPF/UAE w/ Stop Command in Event Procedure & Deleted Sub
- FIX: GPF When Pasting 8 Bit .DIB File into Anibutton Control
- FIX: VB MCITEST CD Player Sample Displays Incorrect Track
- FIX: GPF/UAE After Undoing Edit of Option Explicit Statement

FIX: GPF/UAE When Assign NULL to VBM_GETPROPERTY of type HLSTD

- FIX: Using Graphics Method on DB Objects May Cause GPF/UAE
- FIX: Adding Watch Point in VB May Cause UAE in Windows 3.0
- FIX: GPF/UAE When Large Tag w/ MultiSelect of 30+ Controls
- FIX: Setting Add Watch May Cause Your Program to Reset
- FIX: Setting Add Watch May Cause GP Fault or UAE
- FIX: Painting Problems When FontItalic Set True for Text Box
- FIX: Grid Control Paints Incorrectly When Press PGUP or PGDN
- FIX: GPF/UAE When New Project Loaded After Large Previous Proj
- FIX: No Out of Memory Error Generated with Text Box > 32K
- FIX: Attempting to Refresh Null TableDef Field Causes GP Fault
- FIX: GPF When Using 8514 Driver with Long String in Text Box

FIX: Changing Decimal Separator Causes Load Errors for Form

- FIX: GPF When Making .EXE File If Forms Saved as Binary
- FIX: Bad .MAK File Prevents Display of Make EXE File Dialog
- FIX Large Sub or Function or Module Can Cause GP Fault or UAE
- FIX: GPF/UAE When Create or Use Huge Array w/ Large Elements
- FIX: Error Message: Timeout While Waiting for DDE Response
- FIX: FixedCols Can Cause Paint Problem with Grid Control
- FIX: Problems Calling DoEvents from a Scroll Bar Change Event
- FIX: MAPI: GPF When Attempt to Download 923 or More Messages
- FIX: Extra Chars in Masked Edit Cause Empty InvalidText Box
- FIX: Text Box/Mask Edit in Select Mode If MsgBox in LostFocus
- FIX: Focus Rectangle Remains When Grid Loses Focus

FIX: GPF When Erase User-Defined Array of Variable Strings

- FIX: Loading Proj Gives Err: Custom control 'Graph' not found
- FIX: GPF When Making EXE If Declare Is Missing Lib & DLL Name
- FIX: Resizing MDIForm with UI Does Not Update Height & Width
- FIX: Scroll Bar Thumb Doesn't Do Change Event as It Should
- FIX: Can't Open ODBCADM.HLP Err Msg During Data Access Setup
- FIX: No Menu Event with Maximized MDI Child
- FIX: Mouse Misbehaves After Changing Graph Visible Property
- FIX: OLE Client: Copying Linked Object Gives Err: Can't Paste
- FIX: GPF/UAE with Huge Array Size as Multiple of 64K Bytes
- FIX: Erase Won't Clear Contents of Huge Fixed Array as Variant
- FIX: VB 2.0 Prof Demo Causes Error: Invalid File Format

FIX: Repaint Prob Adding Graphical Control as Child of Graph

Updates Available

Visual Basic for Windows: Bugs, Fixes, & Updates

Unfixed Bugs

Fixed Bugs

Updates Available

- UPD: GP Fault in KRNL286 When Run EXE on 286 or w/ NT on MIPs
- UPD: Data Manager Source Code Available on CompuServe
- UPD: Oracle ODBC Setup and Connection Issues
- UPD: GENERIC Sample Not Provided with Visual Basic
- UPD: New Setup Wizard Available for Visual Basic
- UPD: New XBASE Driver Available That Fixes Several Problems
- UPD: New SETUP.EXE Available for Visual Basic Version 3.0
- UPD: Invalid file format Error When Run VB app's EXE File
- UPD: New Setup Kit Files Available for Setup1
- UPD: New MSCOMM control available
- UPD: New Access Engine MSAJT110.DLL Available
- UPD: List of Updated Files for Visual Basic

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 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

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: 1.00 2.00 3.00
KBCategory:
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 problem with 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: 1.00 2.00 3.00 3.10
KBCategory:
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 ()
   List1.AddItem "["
   List1.AddItem "\"
   List1.AddItem "a"

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

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

   Combo2.AddItem "a"
   Combo2.AddItem "\"
   Combo2.AddItem "["
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: 1.00 3.00 3.10
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 3.00
KBCategory:
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 products listed above when using Windows version 3.0. We are researching this problem and will post new information here as it becomes available.

Additional reference words: 1.00 2.00
KBCategory:
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 products listed above. 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: APrgWindow

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 products listed above in both Microsoft Windows version 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: 1.00 2.00 3.00
KBCategory:
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 problem 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00 errmsg
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 1.00 2.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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 Visual Basic and choose New Project from the File menu (ALT, F, N).

2. Double-click the form or press F7 to bring up the code window.

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

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

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

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

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

```
Sub Form_MouseDown ()
```

```
      Print "Down"; Button
   End Sub
```

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

```
   Sub Form_MouseUp ()
      Print "Up"; Button
   End Sub
```

7. From the Run menu, choose Start.

8. Using the right mouse button, double-click anywhere on the form.

   The output to Form1 should be as follows:

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

9. Press and hold the left mouse button. The output to Form1 should be as follows:

```
      Down 1
```

10. While holding the left mouse button down, double-click with the right mouse button and keep the left mouse button down. The output to Form1 should be as follows:

```
      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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 1.00 2.00
KBCategory:
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 products listed above. We are researching this bug 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.

Reference(s):

"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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: APrgGrap

BUG: 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 versions 3.0 and 3.1

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 products listed above 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
KBCategory:
KBSubcategory: EnvtRun

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 products listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00
KBCategory:
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 products listed above.
We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 MouseUp procedure instead of click, and then call MouseUp when a key is pressed.

STATUS

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

Reference(s):

"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: 1.00 2.00 3.00
KBCategory:
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 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: 1.00 2.00 3.00 3.10
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00

KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsStd

BUG: MDI Child CTRL+INSERT in Properties List Causes UAE/GPF

Article ID: Q80777

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

In Visual Basic version 1.0, the MDI Child custom control may generate an Unrecoverable Application Error (UAE) in Windows version 3.0 or a General Protection (GP) fault in Windows version 3.1 if you highlight the value of the Icon property of an MDI child in the Settings box and repeatedly press CTRL+INSERT.

RESOLUTION

This article does not apply to later versions of Visual Basic. The MDI Child custom control shipped only with version 1.0. Multiple-document interface (MDI) forms are built into Visual Basic version 2.0 and later, making the custom control obsolete.

STATUS

Microsoft has confirmed this to be a bug in the MDI Child custom control supplied with 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

1. In the Visual Basic 1.0 programming environment (VB.EXE), choose Add File from the File menu, and select MDICHILD.VBX from the Files box.

2. Select the MDI Child tool from the Toolbox.

3. Click and drag on the form to place an MDI child control.

4. Select the Icon property from the Properties list box.

5. Select (highlight) the "(none)" entry in the Settings box.

6. Press CTRL+INSERT (you may need to repeat this sequence of keystrokes 10 or more times).

A UAE or GP fault may occur at this point.

Additional reference words: 1.00 2.00
KBCategory:
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 products listed above. 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 recreate 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: EnvtRun

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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: MDI Child Left/Top Property Wrong in Properties Bar

Article ID: Q80907

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When using the MDI Child custom control supplied with Microsoft Professional Toolkit for Microsoft Visual Basic version 1.0, if you set the MDI Child control Left or Top property using the VB.EXE Properties Bar at design time, the value (the coordinate system) reported on the right side of the Properties Bar may change from the value entered. This change is confusing and makes it difficult to determine what the exact Left and Top coordinates for the MDI child window are.

If you set the Left or Top property for the MDI child window, then click the form and then click the same MDI Child window, the Top or Left value in the coordinate system reported on the far right of the Properties Bar will change. On the other hand, the Left or Top property value in the Properties Bar remains as you entered it. Turning off the Align to Grid feature of the Visual Basic environment has no affect.

RESOLUTION

This article does not apply to later versions of Visual Basic. The MDI Child custom control shipped only with version 1.0. Multiple-document interface (MDI) forms are built into Visual Basic version 2.0 and later, making the MDI custom control obsolete.

STATUS

Microsoft has confirmed this to be a bug in MDI Child custom control supplied with Microsoft Professional Toolkit for Microsoft Visual Basic programming system version 1.0 for Windows. This bug was corrected in Microsoft Visual Basic version 2.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. In the Visual Basic 1.0 programming environment, choose Add File from the File menu, and select the MDICHILD.VBX custom control.

2. Click the MDI Child tool in the Toolbox to select it.

3. Click and drag on the form and place an MDI Child control.

4. Select the Left property from the Properties Bar.

5. In the Settings box, enter 500 (a valid number if the ScaleMode property for the form is set to twips).

6. Verify that the coordinate system on the far right side of the Properties Bar indicates 500 (it is the leftmost figure).

7. Click the form.

8. Click the MDI child window, making sure not to move it.

Note: The Left property on the Properties Bar still indicates 500, whereas the coordinate system on the Properties Bar indicates that the left property is 495. This problem occurs even if you change the ScaleMode property for the form.

Additional reference words: 1.00 2.00
KBCategory:
KBSubcategory: PrgCtrlsStd

BUG: MDI Child Control: Large Height/Width Value Not Accepted

Article ID: Q80908

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

When you set either the Height or Width property of a MDI child window to a very large number at design time, the first attempt does not take the number entered, but reverts to the largest acceptable value, which depends on the ScaleMode property and the size of the parent form.

The second attempt to enter the large value is accepted, although the child window does not get any bigger. The largest value you can then enter for either the Height or Width property of the MDI Child control is 10000000000 (1 followed by 10 zeros) before Visual Basic generates an Overflow message. Visual Basic version 1.0 does not trap property values that are too large on the second attempt unless the value is grossly out of range.

RESOLUTION

This article does not apply to later versions of Visual Basic. The MDI Child custom control shipped only with version 1.0. Multiple-document interface (MDI) forms are built into Visual Basic version 2.0 and later, making the MDI custom control obsolete.

STATUS

Microsoft has confirmed this to be a bug in MDI custom control supplied with Microsoft Professional Toolkit for Microsoft Visual Basic programming system version 1.0 for Windows. This bug was corrected in Microsoft Visual Basic version 2.0 for Windows.

MORE INFORMATION

Steps to Reproduce Problem

1. In the Visual Basic 1.0 programming environment, choose Add File from the File menu, and select the MDI.VBX custom control file.

2. Click the MDI Child tool in the Toolbox to select the MDI Child control.

3. Click and drag on the form to place an MDI Child control.

4. Choose the Height property from the Properties list box.

5. Change the value of the Height property to any number greater than 1000000 (1 followed by six zeros). Note the value will then revert to the largest acceptable height for the control. (By default, the ScaleMode property of the form is set to twips. All ScaleMode properties behave the same way.)

6. Repeat step 5 and note that this time the number will be accepted, although the MDI child window will not be affected or change in any way. The value will displayed in mantissa plus exponential form (1.e+007).

   This value is not a valid Height property value for the Height or Width property of the MDI Child control, and Visual Basic will only disallow the entry the first time.

The expected behavior of Visual Basic is to never allow an invalid value for the Height or Width property.

Additional reference words: 1.00 2.00
KBCategory:
KBSubcategory: PrgCtrlsStd

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 products listed above. We are researching this bug 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: 1.00 2.00
KBCategory:
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 products listed above. 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: 3D Command Button Shows Outline when Outline = False

Article ID: Q81951

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 set the Outline property of the Visual Basic 3D Command Button custom control to False, the button appears to have an outline when the button has the focus.

STATUS

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

MORE INFORMATION

The Visual Basic 3D Command Button custom control has an Outline property that can be set to True or False. When the property is set to False, the outline does not appear around the control at design time. At run time, if the button has the focus, the outline appears around the 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 3D Command Button tool appears in the toolbox.

3. Create one 3D Command Button (Command3D1) and one text box (Text1) on the form (Form1).

4. Set the Outline property of Command3D1 to False.

5. Press F5 to run the application. Press TAB to move the focus between the two controls on Form1.

When the Command3D1 button does not have the focus, an outline does not appear. When the Command3D1 button has the focus, an outline appears around the control, even though the Outline property is set to False.

Additional reference words: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. e are researching this bug and will post new information here in the icrosoft 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: 1.00 2.00 3.00
KBCategory:
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, version 2.0
- 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.

WORKAROUNDS

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 products listed above. We are researching this bug 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: 1.00 2.00
KBCategory:
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 problem in 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 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: 1.00
KBCategory:
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 problem with the Graph custom control supplied with the products listed above. 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: Menu Can Cover Top of MDI Child Control If BorderStyle=0

Article ID: Q83858

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

An MDI child window that contains a frame, has its BorderStyle property set to 0 (none), and is maximized, will appear to slip under and be partially masked by the title bar of the parent form when the parent form has a menu.

RESOLUTION

This article does not apply to later versions of Visual Basic. The MDI Child custom control shipped only with version 1.0. Multiple-document interface (MDI) forms are built into Visual Basic version 2.0 and later, making the MDI custom control obsolete.

STATUS

Microsoft has confirmed this to be a bug in the MDI Child custom control supplied with Microsoft Professional Toolkit for Microsoft Visual Basic programming system version 1.0 for Windows. This bug was corrected in Visual Basic version 2.0 for Windows.

MORE INFORMATION

Because an MDI Child control with a BorderStyle of 0 (none) cannot be maximized by the user (there is no Control-menu box for the control), you must use one of two methods to reproduce the problem:

- You can maximize an MDI Child window that has a BorderStyle of 0 (none) by pressing CTRL+BREAK and setting the WindowState to 2 (maximized) in the Immediate window.

  -or-

- Set the MDI Child control WindowState property to 2 in code.

The following example uses the former method.

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 File. In the Files box, select the MDICHILD.VBX custom control file. The MDI Child tool appears in the Toolbox.

3. Click the Toolbox to select the MDI Child control.

4. Click and drag the form to place an MDI Child Window control.

5. Click the Toolbox to select the frame control.

6. Click and drag the MDI Child window to place the frame control. Place the frame at coordinates 0,0 (top left corner of the MDI Child window).

7. Click the MDI child window, choose the BorderStyle property from the Properties bar, and set it to 0 (None).

8. From the Window menu, choose Menu Design Window to add a menu to the parent form.

9. In the Menu Design dialog box, type "Test" on the caption line and again on the CtlName line. Choose the Done button.

10. From the Run menu, choose Start to run the application.

11. Press CTRL+BREAK to break out of run mode.

12. Open the Immediate window, and type the following to maximize the MDI Child window.

```
MDIChild1.WindowState = 2   'The maximized window style
```

Note: The MDIChild1 window will now occupy the whole form, and the frame caption will no longer be visible because it is covered by the menu bar. If there is no menu item, the maximized MDI Child window positions itself correctly.

Additional reference words: 1.00 2.00 MDIChild MDI Child
KBCategory:
KBSubcategory: PrgCtrlsCus

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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. 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: 1.00 2.00 3.00
KBCategory:
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).

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: 1.00
KBCategory:
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 products listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: Common Dialog Custom Controls Don't Display Printer Fonts

Article ID: Q84839

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

The Common Dialog custom control (COMMDLG.DLL) provided with Microsoft Professional Toolkit for Microsoft Visual Basic version 1.0 for Windows will not display printer specific fonts when you use the ChooseFont dialog box. Regardless of the options chosen, the ChooseFont dialog box shows only screen fonts.

The Common Dialog custom control provides a Visual Basic interface to the Windows "common dialog boxes" provided with the COMMDLG.DLL dynamic link library (DLL). Using the Common Dialog control, you can provide a common dialog box to allow the user to select from available fonts. However, even if the proper options are used, the dialog box provided by the Common Dialog control will only display screen fonts, not printer fonts.

WORKAROUND

The only workaround for this problem is to write a DLL routine (using the Windows SDK or equivalent and a language capable of creating Windows DLLs) that in turn calls the COMMDLG.DLL library to display the ChooseFont dialog box. By writing your own DLL routine, you can be certain that the COMMDLG.DLL library uses the correct device context (hDC) for the printer when determining the fonts available. Information on the COMMDLG.DLL library is available as part of the Microsoft Windows 3.1 Software Development Kit (SDK).

STATUS

Microsoft has confirmed this to be a problem with the Common Dialog custom control supplied with 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 as it becomes available.

Additional reference words: 1.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: MDI Child Control Skips Index with Control Array

Article ID: Q87765

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

You can create a control array of MDI children in the Visual Basic environment by copying and pasting an MDI control on to the same form. You will get the error "MDIChild windows are not allowed to nest," when the focus has been set to the previous MDI child before the Paste command. After correctly pasting the next MDI child on to the form, the index of that MDI child will skip a value. In other words, you will not be able to access the MDI child with that Index value.

RESOLUTION

This article does not apply to later versions of Visual Basic. The MDI Child custom control shipped only with version 1.0. Multiple-document interface (MDI) forms are built into Visual Basic version 2.0 and later, making the MDI custom control obsolete.

STATUS

Microsoft has confirmed this to be a bug in the MDI child custom control supplied with the Microsoft Professional Toolkit for Microsoft Visual Basic programming system version 1.0 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. In the Files box, select the MDICHILD.VBX custom control file. The MDI child tool appears in the Toolbox.

3. Click the Toolbox to select the MDI child control.

4. Click and drag the form to place an MDI child window control.

5. Make sure that the MDI child control is highlighted (it should be highlighted by default). Then choose Copy from the Edit menu.

6. Click the MDI child window to select it. Then choose Paste from

the Edit menu.

7. A dialog box will display asking you if you want to create a control array. Choose Yes.

8. The error message "MDIChild windows are not allowed to nest" will appear. Press ENTER to continue.

9. Click the form making sure that the first MDI child is not selected.

10. From the Edit menu, choose Paste to create a control array of MDI children.

Note: The new MDI child window has an Index value of 2. This number should actually be 1 because the first element of the control array is 0. Thus, the Index value 1 is lost and cannot be accessed.

Additional reference words: 1.00 2.00
KBCategory:
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 with products listed above. We are researching this bug 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: 1.00 2.00
KBCategory:
KBSubcategory: PrgCtrlsStd

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 problem with the products listed at the top 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: 1.00 2.00 3.00 7.10
KBCategory:
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 problem 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: 2.00 3.00
KBCategory:
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 problem 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: 2.00 3.00 GPF multiselect
KBCategory:
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 problem 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.

Additional reference words: 1.00 2.00 3.00 4.50 7.10
KBCategory:
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 problem 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: 2.00 3.00
KBCategory:
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 products listed above. 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: 1.00 2.00 3.00
KBCategory:
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 problem 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: 2.00 3.00
KBCategory:
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 products listed above. 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: 1.00 2.00 3.00
KBCategory:
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 bug 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: 1.00 2.00
KBCategory:
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 bug 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: 2.00 GPF
KBCategory:
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 bug 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:

```
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 creates and deletes 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: 2.00
KBCategory:
KBSubcategory: IAPOLE

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 products listed above. We are researching this bug 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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: APrgPrint

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 products listed above. We

are researching this bug 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: 1.00 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

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 products listed above. We are researching this bug 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 23768 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 will start at the X1 and Y1 coordinates given in the MoveTo API call and end 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: 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 3.00 Help
KBCategory:
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 products listed above. We are researching this bug 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: 3.00
KBCategory:

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 products listed above. We are researching this bug 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: 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 3.00
KBCategory:
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 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 for Windows application.

STATUS

Microsoft has confirmed this to be a bug in the products listed above. 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: 1.00 2.00 3.00
KBCategory:
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. Then place a Set statement that refers to the global database / table / dynaset in a meaningful way (for example, Set myds = db.CreateDynaset(...) versus 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. That will take care of the bug.

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 products listed above. We are researching this bug 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: 3.00
KBCategory:
KBSubcategory: APrgDataIISAM PrgCtrlsStd

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 products listed above. We are researching this bug 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: 2.00 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 3.00
KBCategory:
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 products listed above. We are researching this bug 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: 3.00
KBCategory:
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 products listed above. 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: 3.00 errmsg
KBCategory:
KBSubcategory: PrgOther

BUG: Incorrect Result When Multiple Aggregate Functions in SQL

Article ID: Q101256

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 incorrectly shows 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 products listed above. We are researching this bug 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: 3.00
KBCategory:
KBSubcategory: APrgDataIISAM

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 products listed above. We are researching this bug 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: 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

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 products listed above. 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: 2.00 3.00
KBCategory:
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 products listed above. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 2.00 3.00
KBCategory:
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 product listed above. We are researching this problem and will post new information here in the Microsoft Knowledge Base as it becomes available.

More Information:

Additional reference words: 3.00
KBCategory:
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: 3.00 Access JET default update
KBCategory:
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 products listed above. We are researching this bug 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: 3.00
KBCategory:
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 products listed above. We are researching this bug 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
KBCategory:
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, and/or CMDIALOG.VBX.

This problem does not occur when running under Windows for Workgroups version 3.1 in Enhanced mode, because it does not use the file sharing utility SHARE.EXE. It uses its own file sharing utility (VSHARE.386).

WORKAROUND

There are two workarounds for this problem if you need to use the file sharing utility SHARE.EXE. Use either one or the other.

- Set the read-only attribute of the files SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, and CMDIALOG.VBX. This helps to prevent SHARE.EXE from thinking there is a sharing violation when these files are opened by both the Setup Wizard and the compression utility.

or

- Copy the SETUPKIT.DLL, VBRUN300.DLL, COMMDLG.DLL, and CMDIALOG.VBX files 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.

STATUS

Microsoft has confirmed this to be a bug in the product listed above. 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

KBCategory: Tls
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 problem in the Visual Basic version 3.0 Help file.

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:

```
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
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: 3.00
KBCategory:
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 when using the product listed above 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: 3.00
KBCategory: Envt
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 product listed above. We are researching this bug 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: 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

BUG: 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. 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 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
KBCategory:
KBSubcategory: APrgDataODBC

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 products listed above. 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: 3.00 UAE GPF
KBCategory: Envt
KBSubCategory: EnvtDes

BUG: Error 13 (Type Mismatch) & Error 3061 w/ SQL Queries

Article ID: Q105171

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SYMPTOMS

Incorrect SQL syntax can cause either of the following error messages:

- Type Mismatch (Error 13)
- 1 parameter expected but only 0 were supplied (Error 3061)

The problem is with the error messages. The error message should state that the SQL syntax was incorrect.

You may get a 'Type Mismatch' error with a database object on a line of code that contains a SQL statement. You may also get a 'Type Mismatch' error on a .Refresh statement when you are working with a Data control that has a .RecordSource statement containing a SQL query prior to the .Refresh statement.

In some situations, you could get error 3061 (1 parameter expected but only 0 were supplied) instead of error 13 (Type Mismatch).

CAUSE

The SQL syntax is incorrect.

WORKAROUND

Correct the SQL syntax. Workarounds are provided below in the Steps to Reproduce Problem section.

STATUS

Microsoft has confirmed this to be a bug in the product listed above. 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 data control and two command buttons to Form1.

3. Set the following properties for the Data1 control:

Control	Property	Value Set
Data1	DataBaseName	C:\VB\BIBLIO.MDB
Data1	RecordSource	Authors

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

```
Sub Command1_Click ()
   data1.RecordSource = "Select * from authors where author = 4"
   data1.Refresh        '* this gives type mismatch error
End Sub
```

NOTE: If you change the SQL query to the following, you will receive the 3061 error (1 parameter expected but only 0 were supplied):

```
   data1.RecordSource = "Select * from authors where author = brown"
```

The following query corrects the SQL syntax. It should work correctly without giving an error:

```
   data1.RecordSource = "Select * from authors where author = 'brown'"
```

5. Add the following code to the Command2 Click event procedure:

```
Sub Command2_Click ()
   Dim db As database
   Dim ds As dynaset
   Set db = OpenDatabase("C:\VB\BIBLIO.MDB")
   sqlquery1$ = "Select * from authors where author = 4"
   Set ds = db.CreateDynaset(sqlquery1$)'* this gives type mismatch
error
End Sub
```

NOTE: If you change the SQL query to the following, you will receive the 3061 error (1 parameter expected but only 0 were supplied):

```
   sqlquery1$ = "Select * from authors where author = brown"
```

The following query corrects the SQL syntax. It should work correctly without giving an error:

```
   sqlquery1$ = "Select * from authors where author = 'brown'"
```

6. From the Run menu, choose Start (ALT, R, S) or press the F5 key to run the program. Click the Command1 button to get the error. Then restart the program, and click the Command2 button to get the error.

Additional reference words: 3.00
KBCategory:
KBSubcategory: APrgDataAcc

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 product listed above. This problem is corrected as described in the RESOLUTION section above.

Additional reference words: 3.00
KBCategory: Prg
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 product listed above. We are researching this bug 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: 3.00 Hewlett-Packard H-P
KBCategory: Prg
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

To correct this problem, modify the PATH statement in your AUTOEXEC.BAT file to be C:\WINDOWS instead of \WINDOWS.

STATUS

Microsoft has confirmed this to be a bug in the product listed above. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

Additional reference words: 3.00
KBCategory: SetIns
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 product listed above. 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: 3.00
KBCategory: APrg
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 problem in the products listed above. 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")
      ' 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")
      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: 3.00 erase remove how-to create
KBCategory: APrg
KBSubcategory: APrgDataOther

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

SUMMARY

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.

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: 1.00 2.00 3.00
KBCategory:
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

SUMMARY

When using the Circle statement to draw a segment of a circle with a radius of 0, an "Overflow" error incorrectly occurs.

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

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: 1.00 2.00 3.00
KBCategory:
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

SUMMARY

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.

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

This problem does not occur when running under Windows, version 3.1 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.

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: 1.00 2.00 3.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
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

SUMMARY
=======

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. 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: 1.00 2.00 3.00
KBCategory:
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

SUMMARY

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).

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: 1.00 2.00 3.00
KBCategory:
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

SUMMARY

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 further below.

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.

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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: EnvtDes

FIX: Open Project Dialog Misbehaves If Project Dir Deleted

Article ID: Q75519

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you currently have a project loaded in Visual Basic version 1.0 from a network drive and, from a simultaneous DOS session in Windows, you delete that project and its subdirectory, then the next Open Project command will incorrectly default to the nonexistent directory when you reselect the network drive.

CAUSE

The problem is due to the Windows operating environment and is not a problem with Visual Basic 1.0. The problem is in the API function DlgDirList() which is built into Windows versions 3.0 and 3.1.

WORKAROUND

To work around the problem, type the network drive (for example, Z:\ or Z:*.MAK) in the text box to reset to the root directory on the network drive.

STATUS

This problem occurs when using Visual Basic version 1.0 in both Windows version 3.0 and 3.1. However it does not occur when using Visual Basic version 2.0 in either Windows version 3.0 or 3.1. In other words, you will not run into this problem at all in Visual Basic version 2.0.

MORE INFORMATION

Steps to Reproduce Problem

1. In Visual Basic version 1.0, open a project, and save the project in a subdirectory such as Z:\TEST on a network drive.

2. Without exiting Visual Basic, go to the Windows File Manager or the MS-DOS prompt and delete the directory (Z:\TEST) that contains the project on the network drive.

3. In Visual Basic version 1.0, choose Open Project from the File menu. Visual Basic will notice that the subdirectory is gone, so it resets the Open Project dialog box to the startup drive.

4. Click the network drive letter in the Open Project list box. The Open Project dialog box now incorrectly displays the nonexistent subdirectory Z:\TEST as the current directory.

Additional reference words: 1.00 2.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:

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: 1.00 2.00 3.00
KBCategory:
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: 1.00 2.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: EnvtDes

FIX: StretchBlt() Gives UAE/GPF with 256-Color Video Drivers

Article ID: Q77314

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 Unrecoverable Application Error (UAE) in Windows version 3.0 or a General Protection (GP) fault in Windows version 3.1 can occur when a Visual Basic application calls the StretchBlt() Windows API function and uses a Windows 256-color video driver under these conditions:

- The destination object's AutoRedraw property is set to TRUE.
- The destination object's size is an exact multiple of the source object's size.

This problems occurs in both Windows versions 3.0 and 3.1 with Visual Basic version 1.0, but it occurs only in Windows version 3.0 with Visual Basic version 2.0. It does not occur when using Visual Basic version 2.0 with Windows version 3.1; however, changing the size of the picture box may result in a dithered picture.

CAUSE

This problem is caused by the StretchBlt() function. The problem occurs only when the destination object's size is an exact multiple of the source object's size, the destination object's AutoRedraw property is set to TRUE, and you are using a 256-color Windows video driver.

STATUS

Microsoft has confirmed this to be a bug with the Windows StretchBlt() function. We are researching this bug and will post new information here in the Microsoft Knowledge Base as it becomes available.

MORE INFORMATION

Steps to Reproduce Problem

Note that this error occurs only when using a 256-color video driver.

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

2. Place two picture controls on Form1. Load a bitmap into Picture1

by assigning the Picture1.Picture property.

3. Place the following Declare as one, single line of code in the GLOBAL.BAS module:

```
Declare Function StretchBlt% Lib "GDI" (ByVal hDC%, ByVal X%, ByVal Y%,
   ByVal nWidth%, ByVal nHeight%, ByVal hSrcDC%, ByVal XSrc%,
   ByVal YSrc%, ByVal nSrcWidth%, ByVal nSrcHeight%, ByVal dwRop&)
```

4. Place the following code in the Form_Click event procedure, ensuring that the StretchBlt function is placed on one, single line when you type it into the Visual Basic for Windows code window.

```
' Initialize source object's (Picture1) size.
Form1.ScaleMode = 3
Picture1.BorderStyle = 0
Picture2.BorderStyle = 0
Picture1.Height = 64
Picture1.Width = 64
Picture1.ScaleMode = 3    ' StretchBlt uses pixels.
Picture2.ScaleMode = 3
Picture2.AutoRedraw = -1  ' Turn on Autoredraw for destination.

' Set destination object size to a multiple of the source's size.
Picture2.Height = 2 * Picture1.Height
Picture2.Width = 2 * Picture1.Width

hdestdc% = picture2.hDC
X% = 0: Y% = 0
nWidth% = picture2.scalewidth
nHeight% = picture2.scaleheight

' Assign information of the source bitmap.
hSrcDC% = picture1.hDC
XSrc% = 0: YSrc% = 0
nsrcwidth% = picture1.scalewidth
nsrcheight% = picture1.scaleheight

' Assign the constant to the Raster operation (dWrop&).
dwRop& = &HCC0020    'to simply copy

' Enter the following two lines as one, single line in Visual Basic:
ret% = StretchBlt(hdestdc%, X%, Y%, nWidth%, nHeight%, hSrcDC%,
                  XSrc%, YSrc%, nsrcwidth%, nsrcheight%, dwRop&)

Picture2.Picture = Picture2.Image
```

5. Run the program by pressing the F5 key. Click the background of Form1. If you are using a 256-color Windows display driver, a UAE may occur in Windows version 3.0 or a GP fault may occur in Windows version 3.1. After the error, you should exit Windows entirely and restart your computer to avoid subsequent errors.

Additional reference words: 1.00 2.00
KBCategory:
KBSubcategory: APrgOther

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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsStd

FIX: Text Too Narrow with Italic Fonts in Visual Basic Labels

Article ID: Q79117

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 the FontItalic property of a label control is set to True (-1), the text in the caption property is incorrectly formatted into lines that are about half as wide as the label width.

When the FontItalic property is set to False (0), the text is correctly formatted so that each line of text correctly occupies the width of the label.

WORKAROUND

To work around this problem, use any one of the following alternatives:

- Make the label wider so that the text appears as desired.
- Set the AutoSize property on the label to True (-1). Note that this will format the label caption into one line of text.
- Create several labels, one for each line of text.
- Set the FontItalic property to False (0).

RESOLUTION

This problem occurs in both Visual Basic verion 1.0 and 2.0 when using Microsoft Windows version 3.0, but it does not occur when using Visual Basic version 2.0 with Windows version 3.1.

STATUS

Microsoft has confirmed this to be a bug in Microsoft Visual Basic programming system version 1.0 for Windows. This bug was corrected in Microsoft Visual Basic version 2.0 for Windows when using Windows version 3.1.

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. Place a label (Label1) on Form1.

3. Set the label's Caption property to this: This is a line of text.

4. Make the label just wide enough to display the entire caption on one, single line. Make the label tall enough to display three or four lines of text.

4. Set the FontItalic property to True. The caption is incorrectly formatted as two lines.

5. If you reduce the width of the label, it incorrectly formats the lines to about half the new width of the label.

Additional reference words: 1.00 2.00
KBCategory:
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: 1.00 3.00 3.10 NUMLOCK
KBCategory:
KBSubcategory: PrgCtrlsStd

FIX: DDE from Excel to VB Ver 1.0 Uses Up Windows GDI Heap

Article ID: Q80440

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 Excel for Windows, versions 1.x, 2.x, 3.x, and 4.x
- Microsoft Windows, versions 3.0 and 3.1

SYMPTOMS

Microsoft Windows may eventually hang (stop responding) when memory is not released during a dynamic data exchange (DDE) conversation between a Microsoft Visual Basic picture control and a Microsoft Excel for Windows cell range to which it is automatically or manually linked.

Updates to individual Excel cells are not deleted from memory. This problem does not occur when a Visual Basic label control or text box is automatically or manually linked to Excel.

CAUSE

When data in the Excel spreadsheet is constantly updated, resources are consumed and not discarded; as a result, Microsoft Windows version may eventually hang.

RESOLUTION

The memory is not released when either or both the Visual Basic and Excel applications are exited. To release the memory, you must exit from Windows.

STATUS

Microsoft has confirmed this to be a bug in Microsoft Visual Basic programming system version 1.0 for Windows and in Microsoft Visual Basic version 2.0 when it is used with Windows version 3.0. This bug was corrected in Microsoft Visual Basic version 2.0 for Windows when used with Windows version 3.1.

MORE INFORMATION

Using the Heap Walker utility provided with the Microsoft Windows Software Development Kit (SDK), you can verify that unrecoverable memory is being allocated during a DDE conversation between Visual Basic and Excel. Below are the steps necessary to confirm that memory is being lost during a DDE conversation between Visual Basic and Excel:

Steps to Reproduce Problem
---

1. Start an Excel session.

2. Enter data into cells R1C1 through R3C3 (that is, cells A1, A2, A3, B1, B2, B3, C1, C2, C3).

3. Select these cells.

4. Press CTRL+INS to copy the cells onto the clipboard.

5. 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.

6. Add a Picture control (Picture1) to Form1.

7. From the Edit menu, choose Paste Link. A graphic image of the Excel cell range should be displayed in Picture1.

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

9. Run Heap Walker (a tool provided with the Windows SDK).

10. From the Sort menu, choose Module and Label Segments.

11. From the Walk menu, choose GC(-1) And Walk. This command attempts to discard all discardable objects and then display the memory heap.

12. Select all segments labeled GDI by holding down the left mouse button and dragging the mouse downward from the first GDI segment. The window should scroll up as the segments (or lines) are selected. Release the mouse button when the last GDI segment in the block has been selected.

13. With all the segments highlighted, choose the Add! menu to get a total of the segments and bytes used. Make note of these values for comparison.

14. Activate Excel.

15. Change the values in the selected cell range of Excel a number of times.

    Note: By using repeated patterns such as filling the cells with the value 999999, later when using the Show submenu of the Object menu to see the contents of a GDI segment, you can verify that a segment is a leftover from the DDE conversation. Scrolling through the segment, toward the end of the segment you will see the digits you entered on the right side of the display.

16. Repeat steps 12 through 15.

17. Compare the values returned by Add!. The later value should be

    significantly larger.

18. Activate Heap Walker.

19. From the Object menu, choose Show to view the heap owned by the Windows GDI library.

After performing several changes to the data in the range, Heap Walker should report several new segments marked as discardable (a D should appear in the FLG field), but these segments still remain in memory as nondiscardable segments.

Every update to the Excel cells accessed through the DDE link will create a SHARED segment owned by GDI that contains the data. These segments will be approximately equal in size. (The actual size depends on the length of data that was entered in the cells and the size of the selected range.) If you view the contents of these SHARED segments by choosing Show from the Object menu, you will see each data change performed in Excel. The contents of the DDE message will be located around offset 0600 in the segment view mode through Heap Walker.

Reference(s):

Microsoft Windows version 3.0 Software Development Kit

Additional reference words: 1.00 2.00 hot cold
KBCategory:
KBSubcategory: IAPDDE

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: 1.00 2.00 3.00
KBCategory:
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: 1.00 2.00 3.00 gray grey grayed greyed
KBCategory:
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: 1.00 2.00 3.00 H-P HPII
KBCategory:
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
KBCategory:
KBSubcategory: PrgCtrlsCus

FIX: Graph Control's Negative Values Plot As Positive

Article ID: Q81451

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

Negative values used with certain GraphType and GraphStyle properties in the Graph custom control will not appear to be negative, but will instead appear to be positive values. This only occurs in certain graph types where negative data is either invalid or confusing.

The data itself is not changed when this occurs, it is simply graphed as positive values. The values can be shown to still be negative by changing the graph type or style to one where negative data can be shown.

STATUS

This behavior is by design in Visual Basic version 1.0. It was modified in Visual Basic version 2.0 so that the negative numbers are no longer graphed as positive values.

MORE INFORMATION

The GraphType properties that will not show negative data are in Visual Basic version 1.0 are:

- Stacked-bar types (2-D and 3-D)
- Pie charts (2-D and 3-D)
- Gantt charts
- Log/Lin charts

MORE INFORMATION

Steps to Reproduce Behavior 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 GRAPH.VBX custom control file. The Graph tool will appear in the toolbox.

3. Double-click the Graph tool to create a Graph custom control (Graph1).

4. In the Properties Bar, set the following properties for Graph1:

- GraphStyle = 2 (Stacked Bar)
- Height = 2000
- Width = 3000
- NumPoints = 2
- NumSets = 2

Notice Graph1 has two sets of data, with two blue and green bars. Change the values from the random defaults by changing the values for the GraphData property on the Properties Bar. Enter 20, 40, 60, and 40. You must enter these values individually; you cannot enter the values all at once. Graph1 still contains two blue and two green bars. GraphData will cycle back to the first point in the first set. Change its value from 20 to -20. It will appear as though nothing happened; the graph remains the same. If you change the next value, 40, to -80, the bar to the right will grow higher. Now change the GraphStyle from 2 - Stacked to 0 - Default. You will now see the blue bars plotted under the x-axis, reflecting their negative values.

Additional reference words: 1.00 2.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

FIX: Grid: Changing Font Properties Resets ColWidth, RowHeight

Article ID: Q81463

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

Changing the FontBold, FontName, or FontSize property of the Grid custom control at run time will reset the Grid's ColWidth and RowHeight properties to the default values for the newly specified font property. As a result, you may need to reset the ColWidth and RowHeight properties when you change a Font property at run time.

STATUS

This behavior is by design in Visual Basic version 1.0, but it was modified in Visual Basic version 2.0 so that you no longer have to reset the ColWidth and RowHeight properties when you change a Font property at run time.

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). In the Files box, select the GRID.VBX custom control file. The Grid tool will appear in the toolbox.

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

Property	Value
Height	950
Width	2650

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

```
Sub Form_Load ()
   Grid1.ColWidth(0) = 2000
   Grid1.RowHeight(0) = 500
   Grid1.Row = 0
   Grid1.Col = 0
```

```
      Grid1.Text = "Howdy, Cowboy!!!"
   End Sub
```

5. Add the following code to the Grid_Click event procedure.

```
   Sub Grid_Click ()
      Grid1.FontBold = 0           ' Any of these statements will
      ' Grid1.FontSize = 12        ' cause the ColWidth and
      ' Grid1.FontName = "Courier" ' RowHeight to reset.
   End Sub
```

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

When you run the program, the upper left cell is large enough to contain the phrase "Howdy, Cowboy!!!". Notice that clicking the Grid control to change a Font property causes the cell size to reset to its default size, and the phrase "Howdy, Cowboy!!!" is no longer completely displayed.

Additional reference words: 1.00 2.00
KBCategory:
KBSubcategory: PrgCtrlsCus

FIX: VB Graph Custom Control: BottomTitle Text May Disappear

Article ID: Q81950

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 the Height property of a graph made with the Graph custom control is set to less than approximately 1.75 inches (depending on monitor size and type), the text contained in the BottomTitle will disappear from the graph.

CAUSE

This behavior occurs because the Graph custom control chooses the text font automatically based on control size. When the smallest size text font is exhausted, it blanks out the BottomTitle text.

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

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. Place a Graph control on Form1.

4. Size the Graph control to about 3 inches by 3 inches.

5. Change the Graph BottomTitle property to "This is a test." Notice that the title, "This is a test," is displayed at the bottom of the chart.

6. Notice that as you size the form to be shorter, the typeface scales down automatically. Eventually it will disappear off the graph.

This occurs when the Height property is less than approximately 1.75 inches (depending on monitor size and type), regardless of the form's ScaleMode setting.

Additional reference words: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

FIX: Outline Transparent in 3D Button When Outline=False

Article ID: Q82160

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you set the Outline property of a 3D Command Button custom control to False, the outline will be transparent. Unless the 3D Command Button control has the focus, it does not receive all the necessary paint events (such as when the form is minimized then maximized), so it is not painted correctly. Whatever is behind the form containing the 3D Command Button control (usually the Windows desktop) may show through the one pixel area where the outline would normally be when the outline of the control becomes transparent.

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. This bug was corrected in the Professional Edition of Microsoft Visual Basic version 2.0 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 THREED.VBX custom control file. The 3D Command Button tool appears in the toolbox.

3. Add a 3D Command Button control (Command3D1) to Form1.

4. Add another 3D Command Button control (Command3D2) in a different location.

5. Set the Outline property to False for both controls.

6. Minimize Form1.

7. Maximize Form1.

   Note that the control without the focus has a transparent outline until you select the control and it receives the focus.

8. Press F5 to run the program.

9. Minimize, then maximize Form1. This produces the same results.

Additional reference words: 1.00 2.00
KBCategory:
KBSubcategory: PrgCtrlsCus APrgGrap

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)
	cccccccccccccccccccccc	(20 c's)

```
ddddddddddddddddddddd     (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: 1.00 2.00 3.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus APrgGrap

FIX: 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, version 2.0
- 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 products listed above. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

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: 1.00 2.00 3.00
KBCategory:
KBSubcategory: APrgOther

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: 1.00 2.00 3.00 3.10
KBCategory:

KBSubcategory: PrgCtrlsCus

FIX: Toolkit Setup Routine Causes Out of String Space Error

Article ID: Q85155

The information in this article applies to:

- Microsoft Professional Toolkit for Microsoft Visual Basic programming system for Windows, version 1.0

SYMPTOMS

If you try to copy a file using the Setup routines provided with the Microsoft Professional Toolkit for Microsoft Visual Basic version 1.0 for Windows, and the the source file is just under 64K in size, you will receive an "Out of String Space" error.

CAUSE

This error occurs because of the way Visual Basic 1.0 handles variable-length strings.

When Visual Basic makes the following two assignments

```
FileData = String$(32500, 32)
FileData = String$(32767, 32)
```

the data from the first assignment is not removed from memory until the memory space for the second allocation is obtained. Setup has a problem with this because when your file is just under a 64K segment size, there is not enough memory in the segment to make the second allocation because of other local variable length strings.

In the Setup example provided with the Professional Toolkit in the global code module SETUP1.BAS, locate the subprogram CopyFile. About 15 lines from the bottom of this subprogram, find the following assignment:

```
FileData = String$(BlockSize, 32)
```

Because BlockSize is equal to 32767 in this assignment, if the following previous assignment (about six lines up)

```
FileData = String$(LeftOver, 32)
```

was near the 32767 mark, you will get the error discussed above.

WORKAROUND

To avoid this error, add the following line ABOVE the BlockSize assignment:

```
FileData = ""
```

This will free up the string space, thus ensuring the successful completion of the assignment:

```
   FileData = String$(BlockSize, 32)
```

RESOLUTION
==========

This problem doesn't occur in Visual Basic version 2.0 for Windows, which was enhanced with better memory management.

Additional reference words: 1.00 2.00 setup kit
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
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
KBCategory:

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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:

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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:

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

SUMMARY

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.

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 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.

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: 1.00 2.00 3.00 GPF
KBCategory:
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

SUMMARY

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.

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: 1.00 2.00 3.00 GPF
KBCategory:
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

SUMMARY

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 Fault (GPF) (when using Windows, version 3.1) or an Unrecoverable Application Error (UAE) (when using Windows, version 3.0). A GPF 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.

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: 2.00 3.00

KBCategory:
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: 2.00 3.00 multimedia
KBCategory:
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: 1.00 GPF 2.00 3.00
KBCategory:
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
KBCategory:
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: 2.00 3.00 GPF
KBCategory:
KBSubcategory: APrgDataODBC

FIX: Adding Watch Point in VB May Cause UAE in Windows 3.0

Article ID: Q94243

The information in this article applies to:

- Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0
- Windows version 3.0

SYMPTOMS

An Unrecoverable Application Error (UAE) can occur in Windows version 3.0 if you try to add a Watch point for an expression involving a property of the PICCLIP control.

However, performing the same steps in Windows version 3.1 does not cause a general protection (GP) fault.

This problem does not occur when using the Microsoft Visual Basic version 1.0 Professional Toolkit for Windows because version 1.0 does not support watch points.

STATUS

Microsoft has confirmed this to be a bug in the Professional Edition of Microsoft Visual Basic for Windows, versions 2.0 and 3.0. This problem is corrected by using Windows version 3.1.

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, D). Add the PICCLIP.VBX custom control file. This file should be located in the \WINDOWS\SYSTEM directory.

3. Place a Picture control (Picture1) and a PicClip (PicClip1) control on Form1.

4. Add the picture PASTEL.DIB to the picture property of the Picclip1 control. This file should be in the Visual Basic (\VB) directory.

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

```
Sub Form_Load ()
   PicClip1.ClipX = 0
   PicClip1.ClipY = 0
```

```
    PicClip1.ClipWidth = PicClip1.Width
    PicClip1.ClipHeight = PicClip1.Height
    PicClip1.ClipWidth = PicClip1.Width
    PicClip1.StretchX = PicClip1.Width + 10
    PicClip1.StretchY = PicClip1.Height + 10
    Picture1.Picture = PicClip1.Clip
End Sub
```

6. Press the F8 key to single step through the above code. Stop at the second line of code (PicClip1.ClipY = 0).

7. From the Debug menu, choose Add Watch (ALT, D, A). Enter PicClip1.StretchX as the watch expression.

8. From the Add Watch dialog box, choose OK.

9. From the Run menu, choose Continue (ALT, R, C) or press the F5 key to continue running. This will result in a UAE.

Additional reference words: 2.00 3.00
KBCategory:
KBSubcategory: EnvtDes

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: 2.00 3.00 GPF
KBCategory:

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: 2.00 3.00
KBCategory:
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: 2.00 3.00 GPF
KBCategory:
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: 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsStd

FIX: 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, version 2.0

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 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 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: 2.00 3.00
KBCategory:
KBSubcategory: PrgCtrlsCus

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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
KBSubcategory: EnvtDes

FIX Large Sub or Function or Module Can Cause GP Fault or UAE

Article ID: Q95285

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 a general protection (GP) fault in Windows version 3.1 or an unrecoverable application error (UAE) in Windows version 3.0 when making an .EXE file when you approach the maximum size limit of a sub or function or when a form or code module is larger than about 400K.

WORKAROUND

To work around the problem, break up the code in the large sub, function, or module. In most cases, you will need to identify the largest sub, function, or module in your project and work on breaking it up. If this problem persists after breaking up the largest procedure or module, break up other large procedures or modules until you overcome the problem.

STATUS

Microsoft has confirmed this to be a problem in Visual Basic version 2.0 for Windows. This problem was corrected in Microsoft Visual Basic version 3.0 for Windows.

MORE INFORMATION

The problem occurs with a sub or function because Visual Basic attempts to generate more than 64K of code for the sub or function. However, when you run it from the VB.EXE programming environment, it is possible that the amount of interpreted code generated for the sub or function is slightly less than 64K.

The problem occurs with a code module when the compiler attempts to exceed a compiler segment limit.

Because of these problems, it is usually possible to run the program successfully from the operating environment, but you cannot make an .EXE file.

In Windows version 3.0, there is no way to know whether a large procedure or module is the source of the problem. However, in Windows version 3.1, a GP fault occurs at the address 0067:4C4D if a sub or function is too large or at 0067:4C4E if compiler segment limits were exceeded when you attempted to make an .EXE file.

Additional reference words: 2.00 3.00 GPF
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00 blank
KBCategory:
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: 1.00 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 1.00 2.00 3.00 GPF errmsg
KBCategory:
KBSubcategory: EnvtDes

FIX: GPF When Making EXE If Declare Is Missing Lib & DLL Name

Article ID: Q95829

The information in this article applies to:

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

SYMPTOMS

A general protection (GP) fault or unrecoverable application error (UAE) can occur when the EXE is created if you omitted the Lib keyword and DLL name in an external procedure Declare statement in a Visual Basic code module. Unlike Visual Basic for MS-DOS, the Lib keyword and library name are required in Declare Statements for Visual Basic for Windows.

CAUSE

The syntax checker in Visual Basic incorrectly allows a Declare statement to be entered in a Code Module without these two required elements, which allows the program to be run from within the Visual Basic environment. But any attempt to compile the code fails.

WORKAROUND

Because the Sub and Function procedures in a code module are global to other code and form modules, you only need to declare external procedures within a DLL. The syntax for declaring a procedure in a DLL requires that the keyword Lib and the DLL name be used in the Declare statement. If the Declare includes both the Lib and DLL name using the correct syntax, the GP fault can be avoided.

Here are the two possible syntax structures for a Declare statement:

Syntax 1:

```
Declare Sub globalname Lib "libname" [Alias "aliasname" ]
   [([argumentlist])] ' Enter entire Declare on a single line.
```

Syntax 2:

```
Declare Function globalname Lib "libname" [Alias "aliasname" ]
   [([argumentlist])] [As type] ' Enter entire Declare on a single line.
```

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 New Module (ALT, F, M). Module1 is created by default.

3. Add the following code to the General Declarations section of Module1:

   ```
   Declare Function doesntexist ()  ' No Lib keyword or the DLL name.
   ```

4. From the File menu, choose Make EXE File... and press Enter to create PROJECT1.EXE.

Instead of creating a valid .EXE file and returning to the editing environment, an Application Error dialog box appears stating that a GP fault occurred in Module VB.EXE at 006A:0148. If you close the Application Error dialog box and attempt to run the partially created .EXE file may cause a "cannot start application" error message.

Additional reference words: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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
KBCategory:
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: 2.00 3.00
KBCategory:
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: 2.00 3.00
KBCategory:
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
KBCategory:
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: 2.00

KBCategory:
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: 1.00
KBCategory:
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: 2.00 errmsg 3.00
KBCategory:
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
KBCategory: Prg
KBSubcategory: PrgCtrlsCus

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. The post-release version of VBRUN300.DLL is available from the MSBASIC forum of CompuServe in LIB 1 or from the Microsoft Download Service (MSDL) at (206)936-MSDL. Download the file named VBRUN3.EXE (a self-extracting file that contains VBRUN300.DLL) or download the file named VBRUN3.DLL.

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

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 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
KBCategory:
KBSubcategory: EnvtRun

UPD: Data Manager Source Code Available on CompuServe

Article ID: Q99643

The information in this article applies to:

- Microsoft Visual Basic programming system for Windows, version 3.0

SUMMARY

The source code for the Visual Basic Data Manager is available in LIB 1 of the MSBASIC forum on CompuServe in a self-extracting ZIP file named DATMGR.EXE. You may download the source code, modify, or distribute it royalty free.

You can also download the Data Manager source from the Microsoft Download Service (206)936-MSDL. The filename on MSDL is DATAMGR.EXE (a self-extracting ZIP file).

MORE INFORMATION

The source code has been made available because of a press release announcement made by Microsoft that stated that the source code for the Visual Basic version 3.0 Data Manager would be available on CompuServe.

Additional reference words: 3.00 CISFile3.00
KBCategory:
KBSubcategory: RefsProd

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 on your system

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 that 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 on your system

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.)

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
KBCategory:
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 download the GENERIC sample files from the MSBASIC forum on CompuServe in LIB 16 or from the Microsoft Download Service (206)936-MSDL. The filename is GENERC.EXE (a self-extracting file). After downloading the file, run it to obtain the GENERIC sample files.

STATUS

Microsoft has confirmed this to be a bug in the Microsoft Visual Basic programming system version 3.0 for Windows.

Additional reference words: 3.00
KBCategory:
KBSubcategory: Setins

UPD: New Setup Wizard Available for Visual Basic

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 Wizard after Visual Basic version 3.0 was released. The latest bug fix release (version 1.00.0548) of the Setup Wizard (SETUPK.EXE) can be found in LIB 1 of the MSBASIC forum on CompuServe. It is a self-extracting .ZIP file.

You can also download the new Setup Wizard from the Microsoft Download Service (206)936-MSDL. The filename is SETUPK.EXE (a self-extracting ZIP file).

The More Information section below gives the contents of the SETUPWIZ.TXT file. It outlines the bugs that were fixed.

MORE INFORMATION

Below is the contents of the SETUPWIZ.TXT file provided with the fixed version of the Setup Wizard. This file details changes made to SETUPWIZ.EXE after Visual Basic version 3.0 shipped.

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 a 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 (ie MYAPP.EXE) you would get an "Illegal function call in CreateVBSetup1".

1.00.547 Fixed problem where you have 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. NOTE: 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.

PROBLEMS/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. 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.

Additional reference words: 3.00 Update3.00
KBCategory:
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 on CompuServe in LIB 1 of the MSBASIC forum in a file called XBS110.EXE (a self-extracting ZIP file). This driver fixes several bugs documented below. This is the same driver that is provided with Microsoft Access version 1.10.

You can also download the new XBase Driver from the Microsoft Download Service (206)936-MSDL. The filename is XBS110.EXE (a self-extracting ZIP file).

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
KBCategory:
KBSubcategory: EnvtRun

UPD: New SETUP.EXE Available for Visual Basic Version 3.0

Article ID: Q101145

The information in this article applies to:

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

SUMMARY

Several bugs were fixed in SETUP.EXE provided as part of the Setup Toolkit after Visual Basic version 3.0 was released. The latest bug fix release (version 1.00.004) of SETUP.EXE can be found in LIB 1 of the MSBASIC forum on CompuServe in a self-extracting ZIP file named SETUPK.EXE.

You can also download the new SETUP file from the Microsoft Download Service (206)936-MSDL. The filename is SETUPKIT.EXE (a self-extracting ZIP file). You need to run it to get the .ZIP file and then decompress the .ZIP file to get the new version of the Visual Basic SETUP.EXE file.

The More Information section below gives the contents of the SETUP.TXT file. It outlines the bugs that were fixed.

MORE INFORMATION

Below is the content of the SETUP.TXT file provided with the fixed version of SETUP.EXE. This file lists the changes made to SETUP.EXE after Visual Basic version 3.0 shipped.

SETUP.TXT file for the Setup Toolkit SETUP.EXE version 1.00.004

This file lists the changes made to SETUP.EXE after Visual Basic version 3.0 shipped. The version of SETUP.EXE that was released with Visual Basic version 3.0 has a date and time of 28-APR-1993 12:00 am and is 15312 bytes in size. The file contains no version number.

Installation Notes

Copy this version of SETUP.EXE into your SETUPKIT\KITFILES directory over your existing copy of SETUP.EXE.

Distribution Notes

You are free to distribute this version of SETUP.EXE royalty free.

Features/Bug Fixes

The following are the features and bug fixes that were added to SETUP.EXE after VB 3.0 was released. This file will be updated with the latest bug fixes and features any time a new post-release VB 3.0 version of SETUP.EXE becomes available.

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	This problem has been fixed so that you can run SETUP.EXE from a subdirectory. SETUP.EXE

	"Error - Could not open file: <path name> SETUP.LST	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.

Additional reference words: 3.00
KBCategory: Tls
KBSubcategory: TlsSetWiz

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, query on the following words in the Microsoft Knowledge Base:

```
VBASIC and PROF and DEMO and CMDIALOG.VBX and INVALID and FILE and 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 or Download the control from Lib 1 of the MSBASIC forum on CompuServe. You'll find it

in a self-extracting ZIP file named NUGRD3.EXE. You can also download the new grid control from the Microsoft Download Service (206)936-MSDL. The filename is GRID.EXE (a self-extracting ZIP file).

- 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. The grid control is available in a self-extracting ZIP file named NUGRD3.EXE in the MSBASIC forum on CompuServe in LIB 1. You can also download the new grid control from the Microsoft Download Service at (206)936-MSDL. The filename there is GRID.EXE (a self-extracting ZIP file). 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 products listed above. We are researching this bug 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 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
KBCategory:
KBSubcategory: PrgCtrlsStd

UPD: New Setup Kit Files Available for Setup1

Article ID: Q101624

The information in this article applies to:

- Microsoft Visual Basic for Windows, version 3.0

SUMMARY

The following files have been made available in LIB 1 of the MSBASIC forum on CompuServe in a self-extracting ZIP file called SETUPK.EXE:

   SETUP1.FRM
   SETUP1.BAS

You can also download the new Setup Kit files from the Microsoft Download Service (206)936-MSDL. Download and then run the file named SETUPK.EXE (a self-extracting ZIP file).

The contents of the SETUP1.TXT file that comes with these two files is given below.

MORE INFORMATION

SETUP1.TXT File for Setup Toolkit SETUP1.BAS and SETUP1.FRM

This file lists the changes made to files in the SETUPKIT\SETUP1 directory after Visual Basic version 3.0 for Windows shipped.

Installation Notes

Copy the files provided with this README.TXT file into you SETUPKIT\SETUP1 directory.

Distribution Notes

You are free to distribute these files royalty free.

Bug Fixes

The following are the features and bug fixes that were added to the SETUPKIT\SETUP1 files after Visual Basic version 3.0 for Windows was released. This file will be updated with the latest bug fixes and features any time a new post-release version of these files becomes available.

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 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.

Below are the changes that were 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

Additional reference words: 3.00
KBCategory:
KBSubcategory: TlsSetWiz
```

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 control is available in LIB 1 of the MSBASIC forum on CompuServe or from the Microsoft Download Service (206)936-MSDL. In both places, you'll find the MSCOMM control in a self-extracting ZIP file named 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 whose baud rate is lower than 14.4K baud.

One other change that was made. The default property setting for the Interval property was changed from 1000 to 55.

To install the new control, copy the updated version into 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.

Additional reference words: 2.00 3.00
KBCategory:
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 Access engine library MSAJT110.DLL version 1.10.0001 is available on CompuServe in Lib 1 of the MSBASIC Forum in a self-extracting file named MSAJT.EXE. This file can also be found on the Microsoft Download Service (MSDL) at 206-936-MSDL(6735).

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
KBCategory: APrg
KBSubcategory: APrgDataAcc

UPD: List of Updated Files for Visual Basic

Article ID: Q104863

The information in this article applies to:

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

SUMMARY

The following list shows files that were updated after Visual Basic version 3.0 shipped. The list shows the directories where the files are usually located relative to the Visual Basic directory.

File	Purpose
\WINDOWS\SETUPWIZ.INI	Setup Wizard configuration file
\WINDOWS\SYSTEM\BTRV110.DLL	Btrieve ISAM Driver
\WINDOWS\SYSTEM\GRID.VBX	Grid control
\WINDOWS\SYSTEM\ORACLE.TXT	ODBC Oracle Setup and connection info
\WINDOWS\SYSTEM\MSAJT110.DLL	Access database engine
\WINDOWS\SYSTEM\MSCOMM.VBX	Serial communications control
\WINDOWS\SYSTEM\SQLSRVR.DLL	SQL Server driver
\WINDOWS\SYSTEM\SQORA.DLL	ODBC Oracle driver
\WINDOWS\SYSTEM\VBRUN300.DLL	Visual Basic Run-time library
\WINDOWS\SYSTEM\XBS110.DLL	dBASE and FoxPro ISAM driver
CDK\GENERIC\	Sample custom control source. 15 files
SETUPKIT\KITFILES\SETUP.EXE	Setup toolkit
SETUPKIT\KITFILES\SETUPWIZ.EXE	Application Setup Wizard
SETUPKIT\SETUP1\SETUP1.BAS	Setup toolkit
SETUPKIT\SETUP1\SETUP1.FRM	Setup toolkit

You can obtain these updated files from the following sources:

- On CompuServe in forum MSBASIC in LIB 1.

- Microsoft Download Service (206)936-MSDL

For more detailed information on a specific file update, query the Microsoft Knowledge Base using the file name and the word update3.00.

Additional reference words: 3.00
KBCategory:
KBSubCategory: RefsDoc Setins