

INSTANT VISUAL NPL DEVELOPER'S GUIDE

Version 1.0

COPYRIGHT © 1997 Niakwa, Inc.

23600 N. Milwaukee Avenue
Vernon Hills, IL 60061
U.S.A.

PHONE: (847) 634-8700

FAX: (847) 634-8718

E-MAIL: sales@niakwa.com or support@niakwa.com

DISCLAIMER OF WARRANTIES AND LIMITATION OF
LIABILITIES AND PROPRIETARY RIGHTS

The staff of Niakwa, Inc. (Niakwa) has taken due care in preparing this manual. Nothing contained herein shall be construed to modify or alter in any way the standard terms and conditions of the Niakwa Programming Language (NPL) Support and Distribution License Agreement and Warranty or any other Niakwa License Agreement (collectively, the "License Agreements") by which this software package was acquired.

This manual is to serve as a guide for use with the Niakwa-authored suite of NPL products only, and not as a source of representations or additional undertakings by Niakwa. The licensee must refer to the License Agreements for Niakwa product and service representations.

No ownership of Niakwa software is transferred by any of the License Agreements. Any use of Niakwa software beyond the terms and conditions of the License Agreements, without the written authorization of Niakwa, is prohibited.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior written permission from Niakwa, Inc.

Niakwa, Niakwa Data Manager (NDM), Niakwa Programming Language (NPL), Visual NPL (VNPL, Vinny) and Instant Visual NPL (Instant Vinny) are trademarks of Niakwa, Inc.

All other trademarks are the property of their respective holders

Contents

1. Introduction	
1.1 What is Instant Visual NPL?.....	2
1.2 System Requirements.....	3
1.2.1 Developer Knowledge.....	3
1.2.2 Hardware.....	3
1.2.3 Software.....	4
1.3 Software Contents.....	4
1.4 Installation.....	6
2. Quick Start Tutorial	
2.1 Getting Started.....	9
2.2 Required NPL Program Modifications.....	10
2.2.1 Setting up the Instant Vinny Project.....	10
2.2.2 Modifying the NPL Applications.....	12
2.3 Creating the VB Project.....	13
2.4 Creating a VB Form.....	15
2.5 Makean .EXE.....	20
2.6 Starting the Instant Vinny Application.....	20
2.7 Add a Button to Close the Form.....	23
2.8 Adding a Form to “ATDEVICE”.....	26
2.9 Add Enhancements toan Form.....	29
3. Instant Visual NPL Fundamentals	
3.1 Intended Scope of Instant Vinny.....	34
3.2 How Instant Vinny Works.....	34
3.3 NPL Application Changes Required.....	35
3.3.1 Additional Changes to the NPL Program.....	36

3.4	Visual Basic Considerations.....	38
3.4.1	Visual Basic Requirements.....	38
3.4.2	Visual Basic Form Requirements.....	39
3.5	Mixing Instant VNPL with Standard VNPL.....	41
3.6	Examples.....	42
4.	NPL Reference	
4.1	NPL Variables.....	44
4.2	NPL Procedures.....	45
4.2.1	'VnIVCurrForm\$.....	45
4.2.2	'VnIVDisableColor.....	46
4.2.3	'VnIVInit.....	47
4.2.4	'VnIVSelect.....	48
4.2.5	'VnIVShutdown.....	49
5.	Visual Basic Reference	
5.1	Constants.....	52
5.1.1	Standard Keys (strings).....	52
5.1.2	Special Function Keys (integers).....	52
5.2	Subroutines.....	54
5.2.1	VnGoNPL.....	54
5.2.2	VnSendKeys.....	55
5.2.3	VnSendSF.....	56
5.2.4	VnSendStr.....	57
5.2.5	VnSetHost.....	58
6.	Deployment	
6.1	Using the Visual Basic Setup Wizard.....	60
6.1.1	Manually including the NPL files.....	60

Documentation Conventions

This manual uses the following typographic conventions to describe NPL code and constructs:

Example of NPL Conventions	Description
<code>"VnplDev"</code>	Names of NPL modules (programs) are enclosed within quotations in this font.
<code>'VnClose, 'VnCmd</code>	Names of NPL functions and procedures appear in this font, preceded by an apostrophe.
<code>_VnSys, VnPrint\$</code>	Names of NPL constants and variables appear in this font.
<code>PROCEDURE 'NextRecord/PUBLIC ;NPL ; END PROCEDURE 'NextRecord</code>	Code examples written in NPL appear in this font with ;NPL embedded within the code as an explanatory REMark statement.

These typographic conventions describe Visual Basic code and constructs:

Example of VB Conventions	Description
<code>Main, VnDevDef, VnSetCtrl</code>	Names of VB methods, events, functions and procedures appear in bold.
<i>MyProject, Form1, VnplLink, VnplUtil</i>	Programmatic names (not filenames, titles, or captions) of VB objects appear in bold and italics.
<code>BackColor, Tag, Visible, Clipboard Printer, Screen</code>	Names of VB properties and system objects appear with initial letter(s) capitalized.

File, Edit, Tools, Next, Previous, OK, Cancel Next titles and captions of VB menu choices, tab choices, buttons, check boxes and other objects appear in italics.

<pre>Private Sub NextBtn_Click() Rem VB VnCallProc "NextRecord" End Sub</pre>	<p>Code examples written in VB appear in this font with <i>Rem</i> VB embedded within the code as an explanatory <i>Remark</i> statement.</p>
---	---

These other typographic conventions also appear in the manual:

Example of Other Conventions	Description
BOOT.OBJ, DEMOS.NPL, SETUP.EXE, .DLL	Filenames of native Windows or DOS files (including NPL diskimage filenames) and filename extensions appear in this bold font.
F5, TAB, DEL	Names of keys and key sequences appear in small capital letters.
<i>properties, methods, events</i>	In text, italic letters indicate defined terms, usually the first time they occur in the book. Italics also are used occasionally for emphasis.
RETURN	Text you're instructed to type in appears in this font.

CHAPTER 1

Introduction



This chapter introduces Instant Visual NPL (Instant Vinny) to the NPL developer. Topics covered in this chapter include:

- A description of Instant Vinny
- The differences between Instant Vinny and Niakwa's Visual NPL Product
- Instant Vinny system requirements
- Instant Vinny file descriptions
- Installation of Instant Vinny

1.1 What is Instant Visual NPL?

Visual NPL (Vinny) and Instant Visual NPL (Instant Vinny) together solve the following problem: How do you take an existing character-based NPL application into the graphical world?

To **completely** convert that application to Windows, you need two things: Vinny and some time (to design the Visual Basic forms that Vinny connects to your application, and to adjust your application to respond to the “event-driven” way that Windows works)

But most applications follow an 80/20 rule: 80% of the user screens are used rarely, while the remaining 20% are used a lot. In order to speed the delivery of a graphical version of your product to market, we recommend that you follow this plan:

Phase 1: Convert the high-use 20% of your screens to graphical Windows using Vinny. Convert the remaining low-use 80% to “no-frills-but rapid-deployment” graphical Windows using Instant Vinny. Release this version of your application, allowing you to demonstrate and upgrade existing users to a Windows version of your product.

Phase 2: Once you are receiving revenue from Phase 1 (and have satisfied the market need for a Windows version of your application), convert the Instant Vinny screens to Vinny screens. Release this version of your application.

Instant Vinny, therefore, is a tool intended to help NPL developers quickly integrate part or all of character based NPL applications into the graphical Visual Basic environment, with a minimum of time and effort. This is accomplished by adding a small number of changes to the existing NPL application combined with a small amount of Visual Basic form design and coding.

This allows the developer to improve the look and feel of an application in a short period of time, while reducing the overhead knowledge required. Once this is accomplished, the developer can begin to fully explore the Vinny environment while maintaining support for current applications.

To illustrate this, a quick start tutorial is provided in Chapter 2 of this guide. The tutorial provides a step by step project that is easy to follow, allowing you to become comfortable with the concepts of Instant Vinny in a short period of time.

1.2 System Requirements

This section discusses the hardware and software requirements to successfully install and use Instant Vinny.

1.2.1 Developer Knowledge

While Instant Vinny is designed to be a simple tool, developer knowledge of certain subject matter is assumed throughout this guide. The subject areas include:

- A comfortable understanding of NPL and NPL Release IV features (i.e., modules and procedures)
- A basic understanding of Microsoft Visual Basic
- A basic understanding of Visual NPL

In general, if you have comfortably installed Visual Basic and Visual NPL, the tutorial and example programs provided should be enough to provide a rapid introduction to Instant Vinny. Developers should refer to the Visual NPL Developers Guide for details on installing and using Visual NPL.

1.2.2 Hardware

No specific hardware is required beyond that which is recommended for running Microsoft's Visual Basic. However, for best results it is recommended that the product be used in conjunction with a fast video adapter. In addition, you may want to consider using a 17" monitor.

1.2.3 Software

To develop an application in Instant Vinny, the following software needs to be installed on your system.

- MS-Windows Version 3.1 or later (Windows 95 or Windows NT recommended)
- NPL for MS-Windows Revision 4.22.21 or later
- Visual NPL 2.0 or greater
- Microsoft Visual Basic 4.0 (16-bit version)

Note Revision 4.22.21 is a maintenance update to Niakwa's 4.22 field release of NPL. This update is available for download from Niakwa's world wide web site at www.niakwa.com

1.3 Software Contents

Instant Vinny consists of this guide and one diskette containing a compressed setup program and files. Running the **SETUP** program decompresses the files into:

File Name	Description
CUSTOMENU.FRM	Form used in the "Customer Menu" demo
CUSTOMER.FRM	Form used in the "Customer" demo
DATAFORM.FRM	Form Used in the "Data" demo
DEFAULT.FRM	Sample Instant Vinny Form
EXITSCRN.FRM	Instant Vinny Exit Form
HIDESHOW.FRM	Form used in the Hide/Show Demo
ICONINFO.TXT	Describes which Icons you should create in Windows
INSTVNPL.BAS	Instant Vinny Visual Basic library
IVDEMO.EXE	Compiled Instant Vinny Demo Visual Basic Library
IVDEMO.NPL	Instant Vinny Demo NPL Library
IVDEMO.OBJ	Instant Vinny Demo Object File
IVDEMO.VBP	Instant Vinny Demo Visual Basic Project
MAINMENU.FRM	Form used as the Main Menu

NOBORDER.FRM	Form used for the No Border Demo
OKCANCEL.FRM	Form used for the Options Demo
OPTIONS.FRM	Form used for the Options Demo
RTIFORM.FRM	Generic RunTime Form
RTIFORM.FRX	VB compiled form (of above)
SWOOSH.ICO	Icon file used for the Custom Control Toolbar
VNPLDEV.BAS	Visual NPL developers module
VNPLFORM.FRM	Visual NPL connection control form
VNPLFORM.FRX	VB compiled form (of above)
WELCOME.FRM	Opening Instant Vinny Form
WELCOME.FRX	VB compiled form (of above)

1.4 Installation

To install Instant Vinny from the distribution diskette onto your system, run the **SETUP.EXE** program on the diskette. This program does the following:

- Prompts you for the drive and directory to copy the Instant Vinny files into. This should be a subdirectory of your Visual NPL 2.0 directory
- Decompresses the files and copies them to the specified directory
- Creates a new program group of folder depending on which Windows environment is detected
- Creates icons for the Instant Vinny demo program

CHAPTER 2

Quick Start Tutorial



This chapter contains a step by step tutorial demonstrating how to apply Instant Vinny to the standard Niakwa Utility programs distributed with the NPL Development Package.

Upon completion of this tutorial you will have converted the NPL Utilities main menu program from the programs' typical look and feel in Figure 2.1 to the Instant Vinny look and feel illustrated in Figure 2.2.

This chapter discusses:

- Getting started
- Required modifications to the NPL program
- Creating the VB project
- Creating VB forms
- Making an executable for distribution
- Enhancing the NPL program by adding controls to the VB form

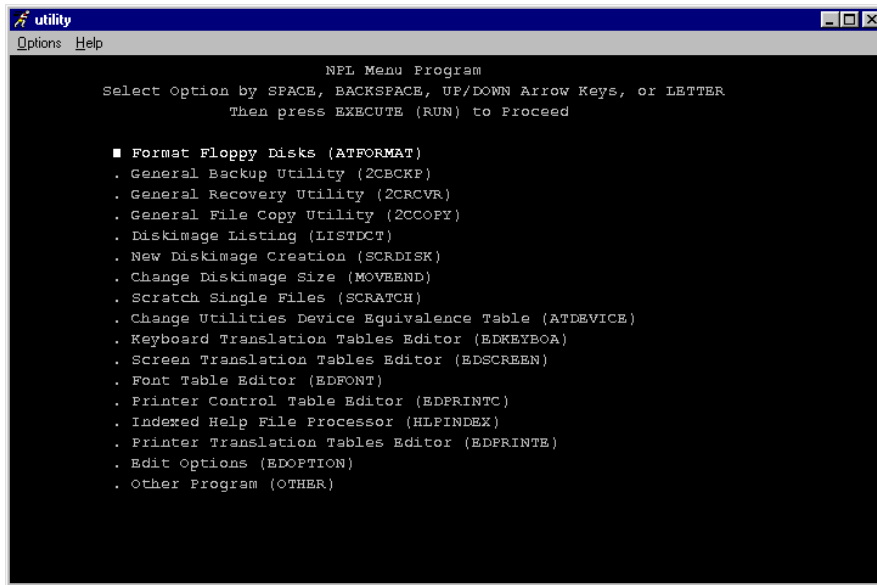


Figure 2.1 Typical NPL Utility Menu

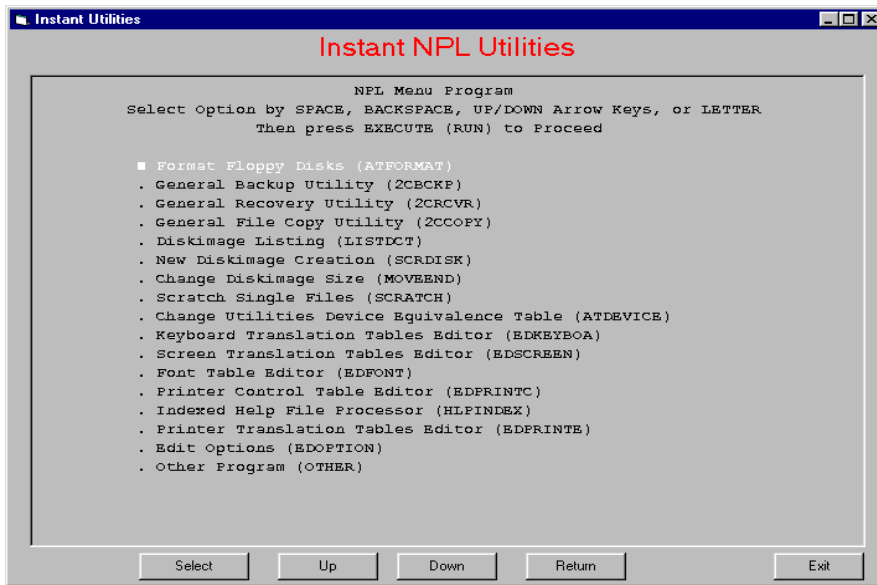


Figure 2.2 Instant Vinny Utility Menu

2.1 Getting Started

Before beginning this tutorial, the following items must be installed and working properly on your system.

- Visual NPL 2.0
- Instant Visual NPL
- The Niakwa Utilities (`UTILITY.BS2` and `UTILITY.OBJ`)
- Visual Basic 4 (16-bit)

Once you're comfortable the above programs are in place and operational, you are ready to begin. The scope of this tutorial is to adapt the standard NPL Utilities Menu program (`2CMENU`) to an enhanced graphical version using Instant Vinny. To accomplish this, the following tasks will be performed:

- Create a working directory and diskimage for your Instant Vinny Project
- Create a boot program for the Instant Vinny application
- Modify two NPL Utility programs to prepare them to use Instant Vinny
- Create a VB project to support the NPL application
- Create an executable program to deliver the application
- Add custom controls to enhance the NPL application

The following sections will guide you through each of the above tasks.

2.2 Required NPL Program Modifications

This section will guide you through setting up files for the Instant Vinny tutorial and modifications required to the NPL programs.

2.2.1 Setting up the Instant Vinny Project

The first step in creating an Instant Vinny project is to create a project directory and copy all required diskimages and programs into this directory. So, assuming you have installed Instant Vinny into a subdirectory of Visual NPL, create and change into the following directory:

```
C:\VNPL20\INSTVNPL>MD IVUTIL
```

```
C:\VNPL20\INSTVNPL>CD IVUTIL
```

```
C:\VNPL20\INSTVNPL\IVUTIL>
```

Once this is accomplished, copy the current NPL Utilities into this directory. Assuming these are in a BASIC2C directory, perform the following:

```
C:\VNPL20\INSTVNPL\IVUTIL>COPY C:\BASIC2C\UTIL* *
```

Note The NPL Utilities are also available on the NPL Utilities Disk included with your NPL Development Package.

The next step to perform is to expand the **UTILITY.BS2** diskimage to accommodate several new programs that need to be added to the diskimage. To do this, use the NPL Utilities to perform the following tasks:

1. Upon startup of the NPL Utilities, add the following diskimages to the NPL Device Table:

C:\VNPL20\VNPL.NPL

C:\VNPL\20\INSTVNPL\IVDEMO.NPL

After establishing the above device references, proceed to the NPL Utilities Main Menu.

2. From the Main Menu, run the "Change Diskimage Size" program and extend the end catalog of the **UTILITY.BS2** diskimage (Device D35) to 1000 sectors.
3. Run the "General File Copy" program and copy the contents of the **VNPL.NPL** diskimage into **UTILITY.BS2**.
4. Run the "General File Copy" program and copy the file "INSTVNPL" from the **IVDEMO.NPL** diskimage into the **UTILITY.BS2** diskimage.

Once the above steps have been completed, the Instant Vinny project is in place and you are ready to begin modifying the NPL programs.

2.2.2 Modifying the NPL Applications

In order to migrate an NPL application to Instant Vinny, the application must be able to locate the Instant Vinny support programs and the name of the file that contains the Visual Basic forms and controls.

2. Load the "2CMENU" program from **UTILITY.BS2** and add the following two lines to the beginning of the program.

```
INCLUDE T#0,"InstVnpl"  
  
'\nIVInit ("IVUTIL")
```

The first line causes the module "InstVnpl" to be loaded, allowing the NPL application to reference all of the Instant Vinny variables and routines.

The second line '\nIVInit ("IVUTIL") , initializes the **IVUTIL.EXE** program. This program will be created later on in this tutorial.

5. In the mainline code of "2CMENU" , you need to specify the name of the Visual Basic form to use. In this particular case we can add this line after the two lines discussed above. Let's call the form "MAIN1" . To do this add the line

```
'\nIVSelect ("MAIN1")
```

3. Resave the "2CMENU" program.

Note Any time you change forms in Instant Visual NPL you must call the '\nIVSelect ("**<form name>**") procedure to associate the new form with the NPL application.

6. Add the same three lines of code to the first line of the "ATDEVICE" program. This is a requirement, since this program is loaded first by the NPL utilities and displays output to the screen before the "2CMENU" program.

2.3 Creating the VB Project

We need to create a Visual Basic Project that contains the forms and code expected by the Instant Visual NPL application. To do this perform each of the following steps.

1. Start the VB4 16-bit version.
2. In the opening screen, from the Visual Basic menu bar, select **File / Add**.
7. In the files selection box, select the directory containing the Instant Vinny files.
8. Add **INSTVNPL.BAS**, **VNPLUTIL.BAS**, **VNPLDEV.BAS**, and **VNPLLINK.FRM** to the project.

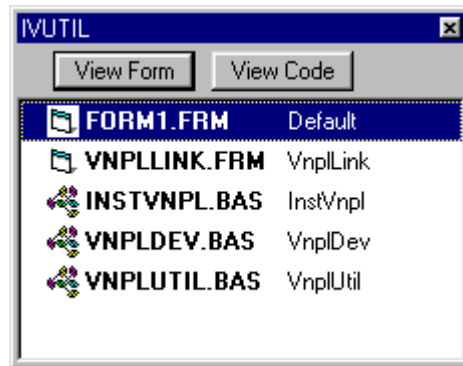


Figure 2.3 Visual Basic Project Window

Note The selected files have now been added to your “Visual Basic project”. These files are required on all Instant Visual NPL projects.

In addition, you will be adding other forms to your Visual Basic project later in this tutorial.

9. Return to the Visual Basic menu bar and select **File/Save Project As**. You may have to click the **OK** button several times until you get to the dialog box requesting the project name. Save this project as **IVUTIL**. The project will be saved as **IVUTIL.VBP**. Later this file will be compiled into **IVUTIL.EXE**, which is the file initialized by the NPL application using the `\nIVInit()` function.
10. Return to the Visual Basic menu bar and select **Tools/Custom Controls**. Scroll down in the **Available Controls** window and check the selection box by **Vnpl Connection Control**. A control icon with a Niakwa swiggle on it should appear on your control tool bar.



Figure 2.4 Visual Basic ControlTool

Upon completing this section, you have successfully created a Visual Basic Project that we will build upon to develop an Instant Vinny version of the NPL Utilities.

2.4 Creating a VB Form

Now that our project is created, the next step is to create a VB Form. This form will represent the NPL Window. When starting Visual Basic, the first form displayed is blank and is typically named **Form1**. Using **Form1**, perform the following tasks:

1. Click on the form to assure it is in focus.
2. Change the Name property in the Properties Window to **"MAIN1"**.
3. Change the Caption property in the Properties Window to Instant Utilities.

Note It is highly recommended that all forms have the Maximize property set to False and the BorderStyle property set to 1 (Fixed Single). These settings prevent forms from being either Maximized or Resized. Since it is awkward to reposition buttons and size the NPL window accordingly, you do not want the forms to be resized.

4. Select the **3-D Text Box** from the **VB Controls Tool Bar**. Any text box will do, but the **3-D Text Box** just looks nicer. Make the text box almost as large as the form.

Note Change the Name property of a form before adding controls to the form. Otherwise, the controls will be lost when you rename the form.

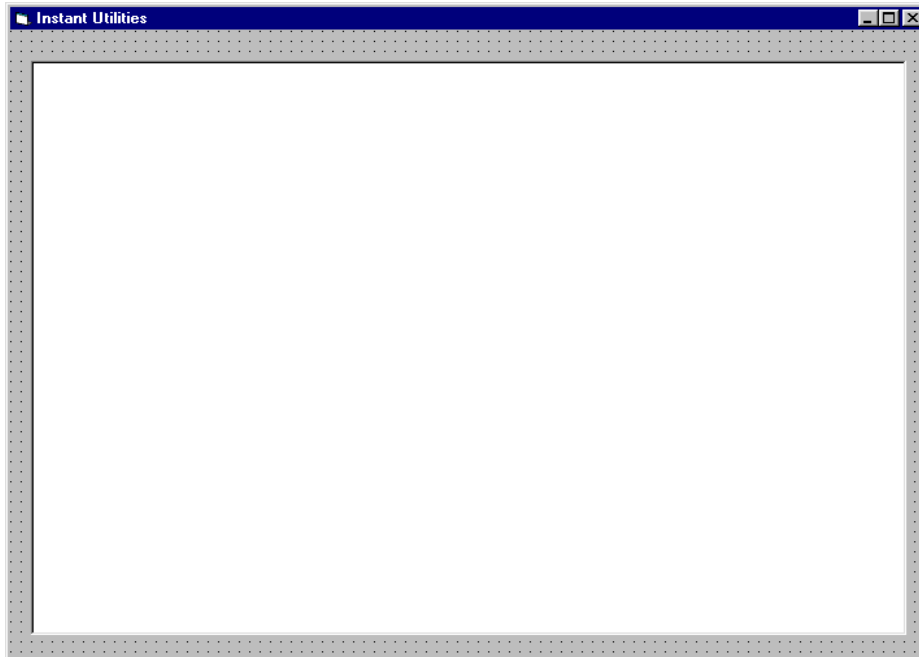


Figure 2.5 Main1 form with 3-D text box applied

5. After adding the 3-D text box control to the form, change the Name property of the 3-D text box control to **RTIWINFrame**. Any name is acceptable, but we recommend using **RTIWINFrame**. This naming convention helps make your code easier to debug.

Note Always change the Name property of a control before adding any code to the control. Otherwise, the code will be lost when you rename the control.

6. Double click on the form itself (not the 3-D text box) to bring up the code box for the form.
7. In the FORM_LOAD() function, add the following line as displayed in Figure 2.6. The `VnSetHost` statement ties the NPL window to the control named **RTIWINFrame**.

VnSetHostRTIWINFrame

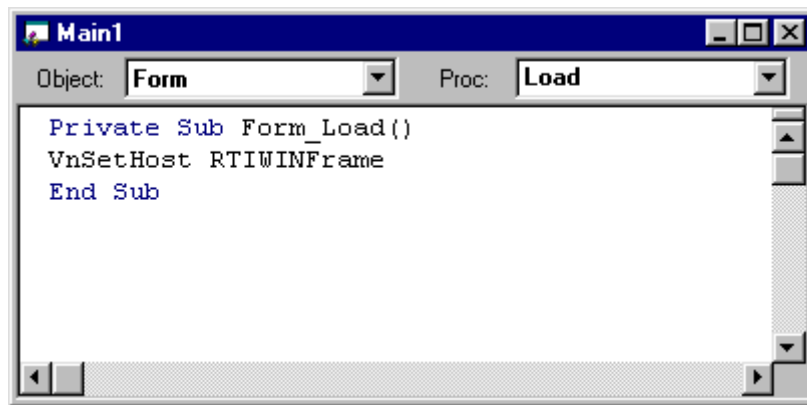


Figure 2.6 Load Form Command Window

8. To prevent the VB form from closing and leaving RTIWIN "floating" on the desktop (without a title bar or border), the QueryUnloadevent should be trapped on all forms. It is best to handle the closing of the VB forms by sending the appropriate keystrokes back to the runtime and having it exit the NPL application so as to shut everything down properly. In the code box for the **Main1** form, in the QueryUnload() procedure enter:

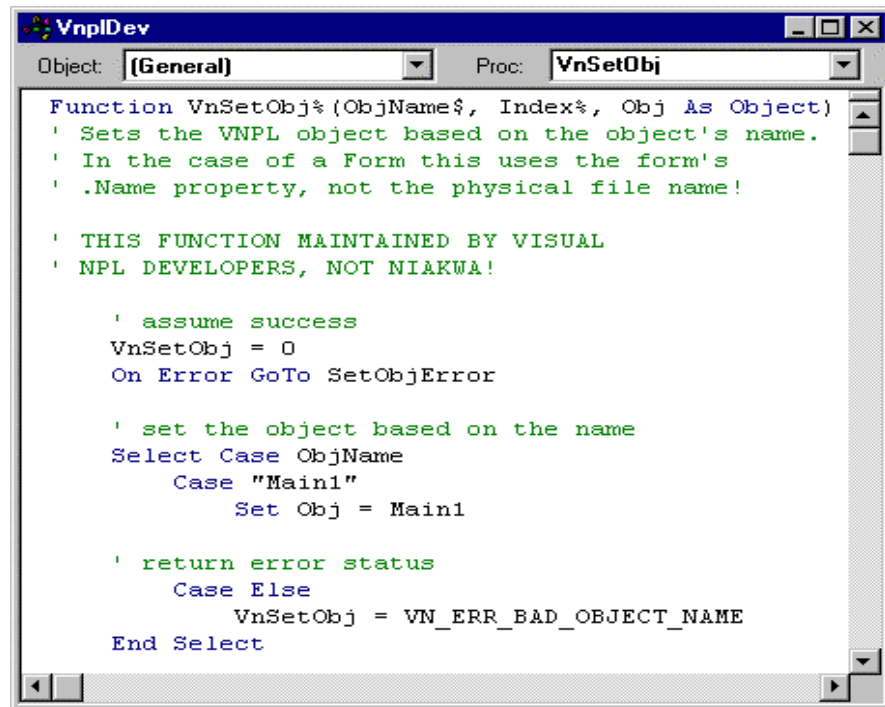
```
If UnloadMode < > 1 Then
    Cancel = 1
    '
    'if typing the letter "Y" ends the
    'application, then use the following
    'line
    'VnSendStr "Y"
    '
End If
```

Note If you want the closing of a form to quit the NPL application , you should send the appropriate keystrokes to the NPL application to have it shutdown. The keystrokes would be the same that a Close button would send to the runtime.

Our next step is to add the name of our new form to the ~~select~~ statement in **VNPLDEV.BAS**. To accomplish this:

1. Double click on **VNPLDEV.BAS** in the project window.
2. The top right window will have "[declarations]" in it. Click on the down arrow next to the window and a selection box will drop down. Select **VnSetObj**.

- The code for `VnSetObj` will appear in the main window. Scroll down to the `Select Case ObjName` statement. There should be several entries in that statement already. Using the existing entries as an example, enter the name of your newly created form (**MAIN1**) to the statement.

The image shows a screenshot of the VnplDev IDE. The window title is "VnplDev". The "Object:" dropdown menu is set to "[General]" and the "Proc:" dropdown menu is set to "VnSetObj". The main text area contains the following VBA code:

```
Function VnSetObj%(ObjName$, Index%, Obj As Object)
' Sets the VNPL object based on the object's name.
' In the case of a Form this uses the form's
' .Name property, not the physical file name!

' THIS FUNCTION MAINTAINED BY VISUAL
' NPL DEVELOPERS, NOT NIAKWA!

' assume success
VnSetObj = 0
On Error GoTo SetObjError

' set the object based on the name
Select Case ObjName
    Case "Main1"
        Set Obj = Main1

' return error status
    Case Else
        VnSetObj = VN_ERR_BAD_OBJECT_NAME
End Select
```

Figure 2.7 Select Case statement in *VnplDev*

Note Every form in your Instant Vinny application must be identified in this `Select` statement.

2.5 Make an .EXE

The final step in creating the Instant Vinny Utilities is to make a VB executable to launch the Instant Vinny Application. To do this, perform the following steps:

1. From the Visual Basic menu bar select **File / Make An Exe File**
The dialog box will ask where to save the file and what to name it. The default name is the project name. Use the default name and save it in the BASIC2C directory.
2. When you make changes to the project you must resave the project and remake the **.EXE** file. If an **.EXE** file exists with the same name you will be asked if you want to replace the file. (Answer yes).

2.6 Starting the Instant Vinny Application

The Instant Vinny application is started in the same manner as a VNPL application with one addition, you must add the /C parameter on the command line:

```
RTIWIN [<bootfile>] /C /Xvnpl16.dll
```

To start your application, enter:

```
C:\BASIC2C\INSTVNPL\IVUTIL\RTIWIN UTILITY /C  
/XVNPL16.DLL
```

Note The NPL /C command line option is supported by NPL revision 4.22.21 or greater.

The first screen you see is the “NPL Device Configuration” screen applied to the Instant Vinny form.

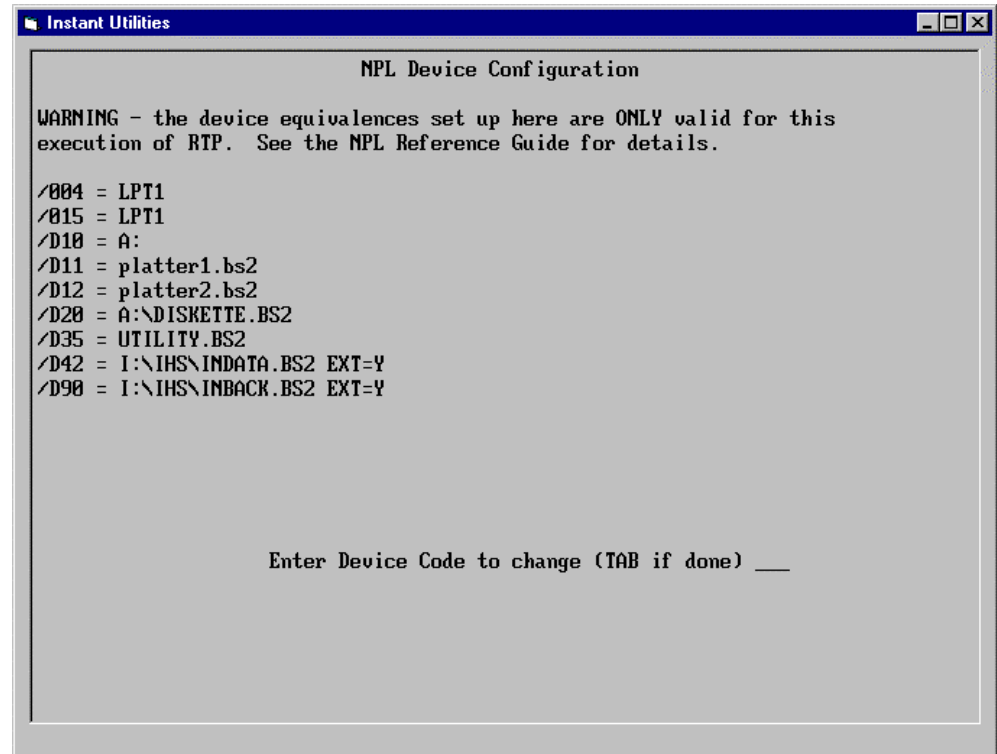


Figure 2.8 Device configuration screen applied to Vinny form

Press the TAB key and do not accept the changes to advance to the next screen.

You should be looking at the familiar NPL Utilities menu applied to the Visual Basic form you just created.

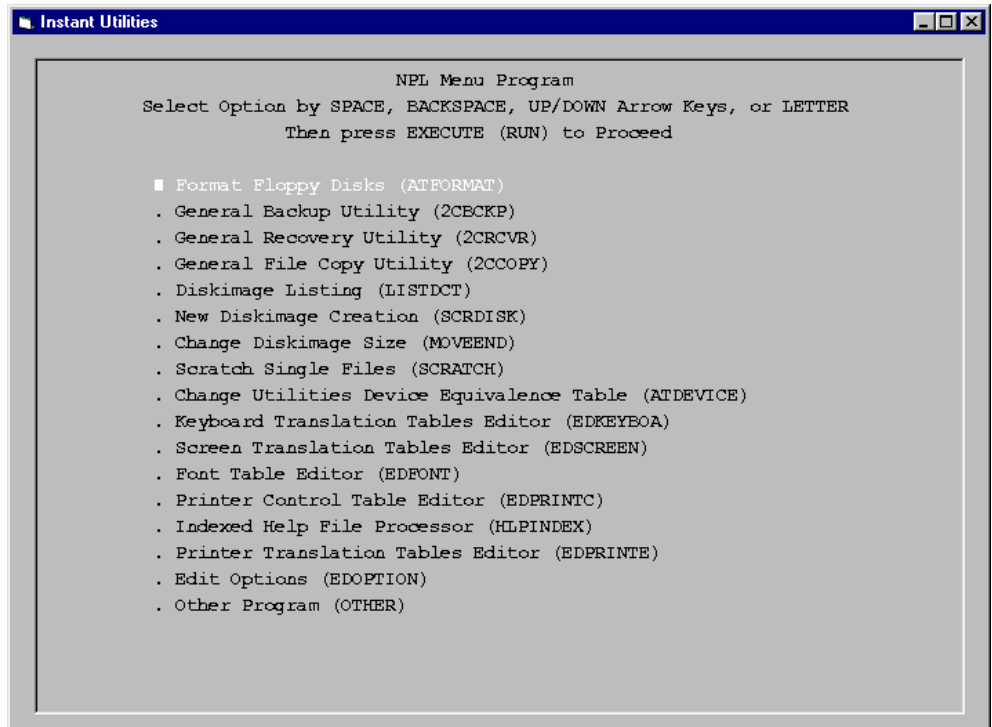


Figure 2.9 Main form with NPL Utilities in 3-D text box

You should now see the modified "2CMENU" program displayed. Other than the display of the menu, the program operates the same as it did when running outside of Instant Vinny.

Exiting from NPL Utilities leaves you with the VB form and an active RunTime screen pasted on it. Enter **END** to exit the RunTime and close the VB form.

In the next section, we will look at dressing up our form a bit.

2.7 Add a Button to Close the Form

This section walks you through adding a command button control to the Instant Utilities application.

1. Start Visual Basic and open your project.
2. View the **MAIN1** form.
3. Click on the **RTIWINFrame** control and resize it to make some room at the bottom for a command button.
4. Drop a command button (from the VB Controls Tool Bar) onto the bottom of the **MAIN1** form and adjust the size so it looks good to you.
5. Click on the new button control and change the Name property of the command button to BtnExit.
6. Change the Caption property of the command button to Exit.

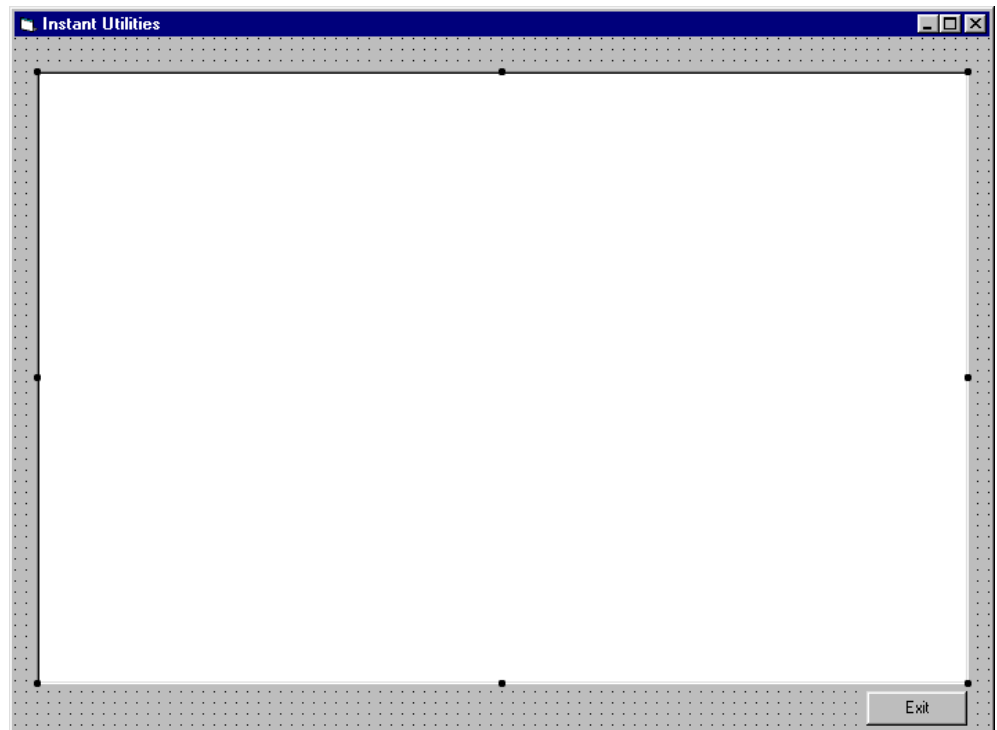


Figure 2.10 Main form with Exit button added

7. Double click on the command button to bring up the code window for it. In the code for the `BtnExit_Click()` function enter:

```
VnSendSF NPL_Cancel
```

```
VnSendStr "$END" & NPL_Return
```

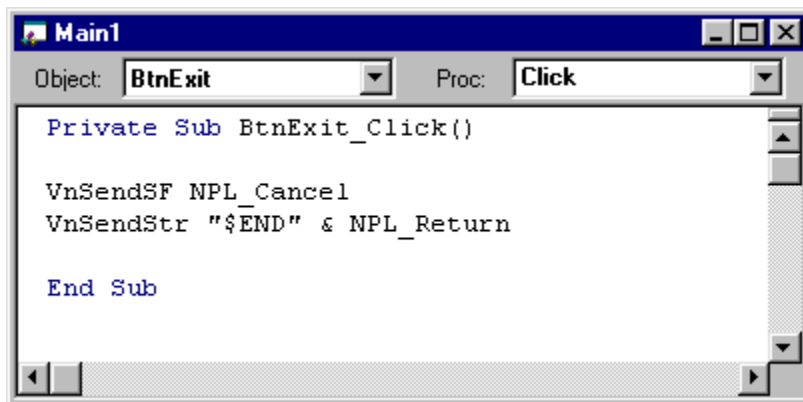


Figure 2.11 Exit Button code screen

Note You could have used the `VnSendKeys` procedure instead of `VnSendSF` and `VnSendStr`. `VnSendSF` or `VnSendKeys` can be used to send special function keystrokes to the NPL application. Special function keys must always be sent singly no matter which procedure is used. `VnSendKeys` and `VnSendStr` can send character strings and standard keys. Several character strings and standard keys can be combined by putting an ampersand (&) between them. They can be sent using either `VnSendKeys` or `VnSendStr` procedure. A character string is always enclosed in quotes.

8. Recompile `IVUTIL.VBP` into an `EXE` file again, replacing the old `IVUTIL.EXE` with the new one.

9. Start the application again by entering `G:\BASIC2C\RTIWIN UTILITY /C /XVNPL16.DLL`. This time there should be an exit button on the bottom of the NPL Device Configuration form and the Utilities Menu form.



Figure 2.12 Instant Visual NPL screen with Exit Button

With the NPL Device Configuration screen on the form, the **Exit** button will not cause the form to close or exit. The reason for this is the "ATDEVICE" program does not understand the commands being input from the **Exit** command button.

Use the keyboard and advance to the NPL Utility Main Menu and clicking the **Exit** command button will cause the form and NPL window to close and exit the RunTime.

The next section will discuss adding a unique form for the "ATDEVICE" program.

2.8 Adding a Form to “ATDEVICE”

This section discusses adding a new form to be displayed when the initial Device Configuration screen of the Instant Vinny Utilities is displayed.

1. If Visual Basic is not running, start it and open your project.
2. Add a new form to your project. From the menu bar select **Insert / Form**
3. Change the Name property of the form to DevFrm.
4. Add a 3-D Text Box to the form. Size it to allow room at the bottom for a command button and at the top for a heading.
5. Change the Name property of the text box to **RTIWINFrame**.
6. Add a command button on the bottom of the form, size it, and change the Name property to BtnCont.
7. Change the Caption property to Continue.
8. In the code for the **BtnCont** command button, enter:

```
VnSendKeys NPL_Tab
```

```
VnSendKeys NPL_execute
```

9. Next, let's add a heading to **DevFrm**.
10. Select a Label control (from the VB Controls Tool Bar) and drop it on the top of the form.
11. Size the Label control to fill the space between the Menu bar and the top of the **RTIWINFrame** control.
12. Center the heading on the form by changing the Alignment property. To change this property, mouse click on the Alignment property to highlight it. A down button appears. Click on the down arrow and a list of options will drop down. Choose **2-Center** with a mouse click to center the heading.

13. Change the color of the text in the heading on the form to red by changing the ForeColor property in the same manner as above. Mouse click on the red color swatch.
14. Change the Font property by selecting it with a mouse click. When you select the property a font selection dialog box will appear. Choose a nice looking font and set the size to 24.
15. Change the Caption property. To do this, select the property with the mouse. Click on the right hand column opposite the Caption property. Now you can enter the text of the heading. Enter **NPL Device Configuration** You will see the text, in red, on the top of the form where you placed the Label control.

Notice how the caption changes as you make changes in the property for that Label. This allows you to see what the changes look like before you actually run the program.

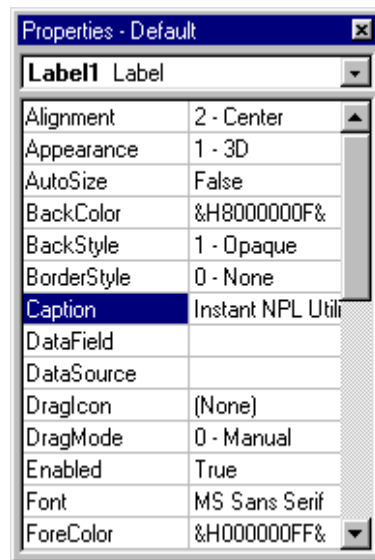


Figure 2.13 Visual Basic Properties Window

The name of the newly created form has to be added to the Select statement in `VNPLDEV.BAS`.

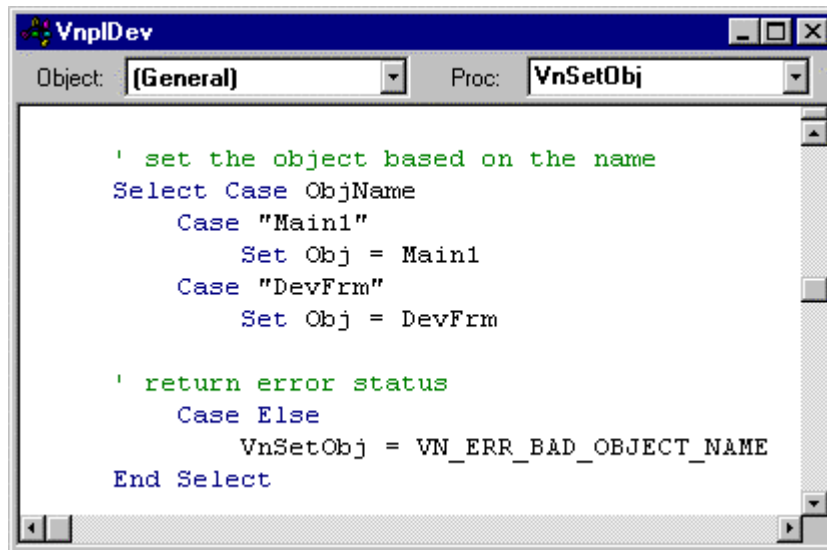


Figure 2.14 Add the form name to the `Select Case` statement in `VnplDev`

Every form in your Instant Vinny application must be identified in this Select statement.

The `'VnIVSelect` statement in the "ATDEVICE" program has to be modified to load the new form.

Change:

```
'VnIVSelect("Main1")
```

to

```
'VnIVSelect("DevFrm")
```

Re-make the EXE and run the program. This time you should be greeted by a new screen with the caption "NPL Device Configuration", the NPL Device Configuration screen, and a button labeled Continue. Press the Continue button and the **Main1** screen will appear with the NPL Utility menu.

In the next section we will modify the **Main1** form and dress it up a bit by adding a header.

2.9 Add Enhancements to an Existing Form

This section discusses how to add a colorful heading to the existing **Main1** form.

1. If Visual Basic is not running, start it and open your project.
2. Reduce the size of the **RTWINFrame** control to allow some room at the top of the form for the heading.
3. Take the Label control from the VB Control Bar and drop it on the top of the form. Size the control to fill the space between the Menu bar and the top of the **RTWINFrame** control.
4. Center the heading on the form by changing the Alignment property. To change this property, mouse click on the Alignment property to highlight it. A Down button appears. Click on the Down arrow and a list of options will drop down. Choose 2-Center with a mouse click to center the heading.
5. Change the color of the text in the heading on the form to red. Change the ForeColor property in the same manner as above. Select the red color.
6. Select a font for the heading. Change the Font property by selecting it with a mouse click. When you select this property a font selection dialog box will appear. Choose a nice looking font and set the size to 24.

- The Caption property is where the text for the heading is actually entered. To do this, select the Caption property with a mouse click. Click on the right hand column opposite the Caption property. Now you will be able to enter the text for the heading. Enter "Instant NPL Utilities". You will see the text, in red, on the top of the form where you placed the Label control.

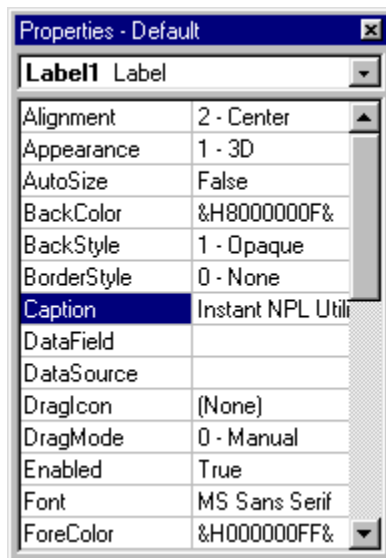


Figure 2.15 Visual Basic Properties Window

8. Re-make the EXE file and start your Instant Visual NPL Utilities.

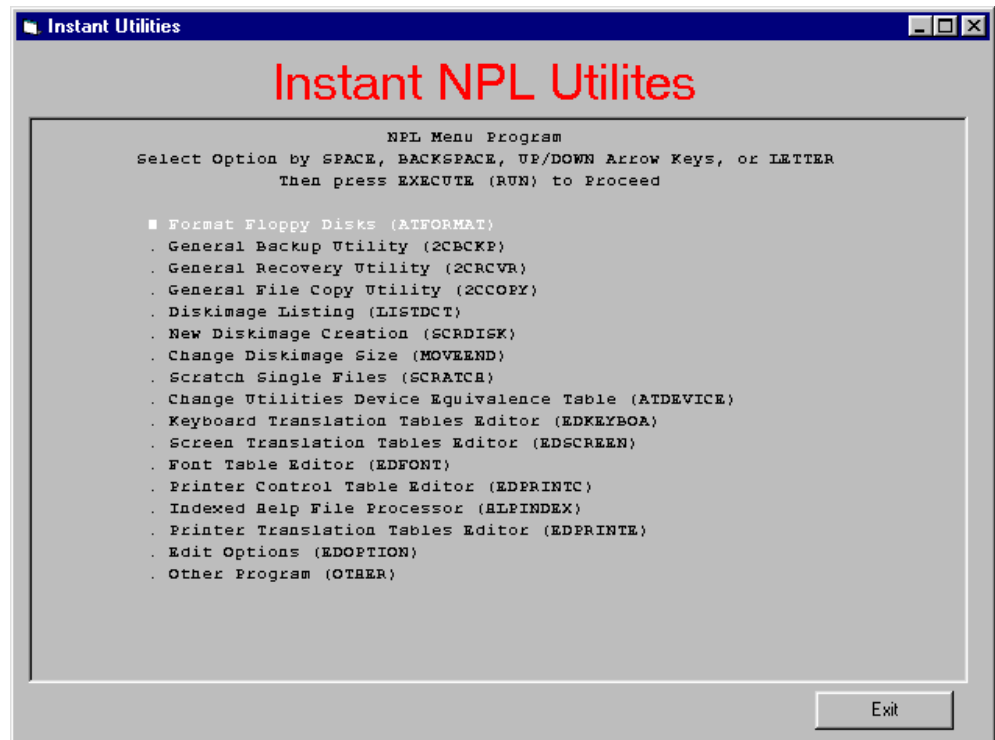


Figure 2.16 Instant Visual NPL form with Heading added.

Congratulations, you have just created your first Instant Visual NPL application.

With just a few modifications to the NPL code and some minor coding in Visual Basic, you have created a Windows look for the NPL Utility menu.

You can add more command buttons to the form, sending different NPL_keys to the NPL application.

For instance:

Create **Up** and **Down** command buttons and send NPL_NORTH and NPL_SOUTH to the application with them.

Make a **Select** button and send NPL_EXECUTE with it.

Now you can navigate the entire main menu screen using only the command buttons on the Visual Basic part of the form.

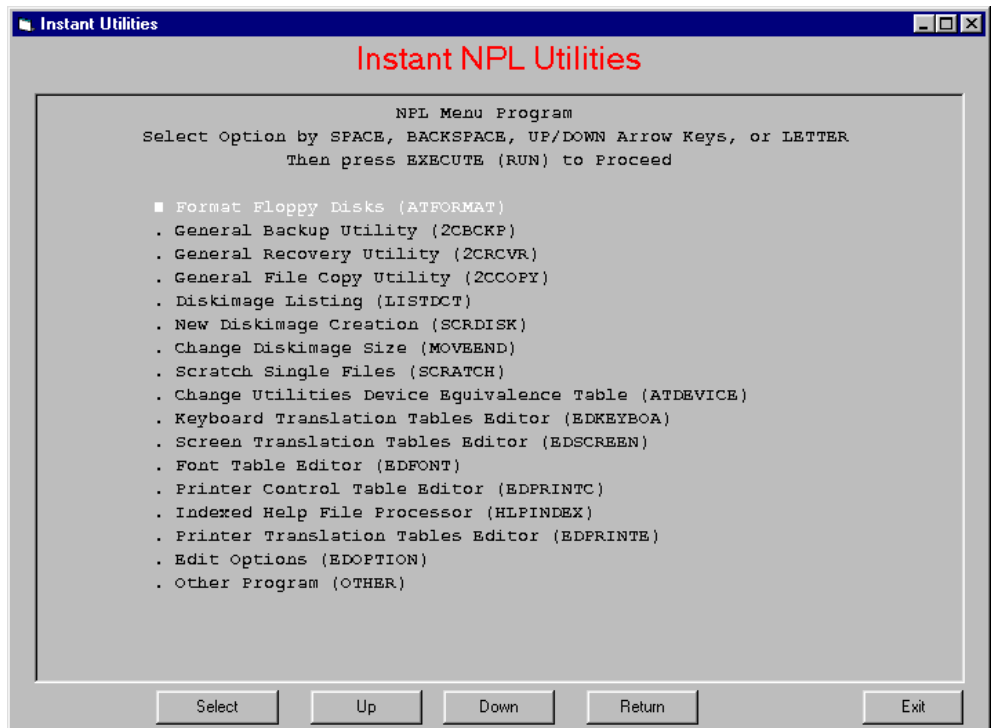


Figure 2.17 Final Instant Visual NPL form

CHAPTER 3

Instant Vinny Fundamentals



This chapter discusses the fundamentals of using Instant Vinny and covers the following topics:

- Intended scope of Instant Vinny applications
- How Instant Vinny works
- Changes required to adapt an NPL application to use Instant Vinny
- Visual Basic requirements
- Visual Basic form design requirements
- Merging an application with both Visual NPL and Instant Vinny routines

Before reading this chapter, developers are encouraged to walk through the “Quick Start Tutorial” in Chapter 2.

3.1 Intended Scope of Instant Vinny

As discussed in Chapter 1, Instant Vinny is a development tool designed for the specific purpose of improving the overall look and feel of an NPL application in a Windows environment. Instant Vinny is designed to be simple to use and implement.

The intention of this product is to provide developers with a simple path to integrate an NPL application with Visual Basic without having to be an advanced Visual Basic or Visual NPL user.

3.2 How Instant Vinny Works

An Instant Vinny application is an NPL program running in a special RTIWIND window that appears to be “pasted” on a Visual Basic form. This special NPL window has no border, no title bar, and no menu. It looks like a type of Visual Basic control.

Note This is especially true if the application’s color schemes can be modified to closely match those commonly used in Windows. Techniques to do this are found later in this guide.

The Visual Basic form is usually quite simple, and is designed to allow push and/or radio buttons to emulate normal and SF keystrokes typed in the NPL program. Normal or Special Function key combinations used by the NPL application are executed by clicking on an appropriate button located on a Visual Basic form, in addition to pressing keys from the keyboard as always. Combining these buttons with picture boxes and other display objects enhances the appearance of the application. The behavior and performance of the NPL application, however, remains unchanged.

3.3 NPL Application Changes Required

In order to adapt an NPL application to use Instant Vinny, several changes to the NPL program are required.

The first required change is to insert a line of code that will point the NPL application to the Visual NPL libraries. For example, the first line of your NPL application should be:

```
INCLUDE T "InstVnpl"
```

Next, a line of code is added that initializes Visual NPL, and loads the compiled VB program containing the Visual Basic form(s) you will be using. For example:

```
'VnIVInit("MyApp")
```

The above command initializes Visual NPL using the `'VnIVInit` procedure, and loads `MyApp.EXE`.

Note The VB program may or may not exist at this point. The code just needs to reference the correct name of a VB application at runtime.

Once the application and VB are initialized, the NPL window remains hidden by default. To make the form visible, use the `'VnIVSelect` procedure to load a Visual Basic form. For example,

```
'VnIVSelect("MAIN1")
```

Note Add this statement after the application's code initialization section.

After the VB form has been loaded, the NPL window will appear to be pasted onto the VB form, very much like a standard VB control. The NPL application may switch VB forms at anytime by issuing a subsequent `'VnIVSelect` procedure and specifying a different VB form name.

At this point, the NPL application will execute just as it did before. In addition, the application exits just as before and Instant Vinny will automatically shut itself down.

The changes discussed above are the minimal code requirements needed to initialize Instant Vinny and load a single Visual Basic form. However, most programs have a variety of screens in which the SF key functionality may change. Therefore it may be necessary to switch VB forms occasionally to assure that the keystrokes associated with button controls on the VB form are logically consistent with the keystrokes your application is expecting. In any case, the NPL application will continue to respond to keystrokes entered from the keyboard as normal.

The number of VB forms an application uses is up to the developer. The Instant Vinny approach makes virtually all of the VB forms similar in nature so they can be duplicated and then changed in a relatively short period of time.

The number of forms an application uses can be held to a minimum by 'overloading' the form with more than the required controls. Thus, instead of changing forms every time you switch NPL screens, a single form can represent many forms by making controls visible and invisible upon demand without changing forms. Examples of this are included in the Instant Vinny demo application and VB forms.

3.3.1 Additional Changes to the NPL Program

To make your application look as "Windows-like" as possible, it is advisable to modify the NPL color scheme to match the standard Windows display colors as closely as possible. RTIWIN does part of this by default by mapping white to the standard Windows foreground color and black to the standard Window's background color.

The most common color scheme for VB forms (and Windows applications in general) is black text on a gray background. This contrasts rather poorly with most color schemes used by typical text character based NPL applications. A fairly common text application will use white or bright white on blue, which will appear quite abnormal pasted onto a standard VB form.

Changing the VB colors would make the entire application look different from standard Windows applications, which would defeat

a primary purpose of Instant Vinny. Therefore it is suggested that the NPL application be modified to take on the standard color scheme of Windows applications if multiple color schemes are used.

This may not be practical if the NPL application changes colors without using a library of procedures that define default foreground and background colors or override certain or all color combinations. Furthermore, certain colors might be used to highlight data fields, push buttons, or other regions of the screen that would otherwise be difficult to see if the color schemes were modified. The developer should take care in re-mapping the application's colors so that functionality is not lost.

The default colors used for button background and text are determined as follows:

1. If the RTIW.INI settings for StandardRGBColor0 and StandardRGBColor7 are assigned values, these color values are used for NPL text where the program specifies black and white color, respectively.
2. In the absence of RTIW.INI settings, the default NPL Window colors of black and white are replaced by the current desktop color values for button background and text respectively.

In addition, Instant Vinny provides the procedure 'VnIVDisableColor' which may assist in standardizing the look of your NPL application. Another possible method is to remap how RTIW.INI displays colors by using the color settings feature of RTIW.INI. For example, the following would be set in the GENERAL section of RTIW.INI :

```
[GENERAL]
StandardColorRGB0=192 192 192 ;black to gray
StandardColorRGB1=192 192 192 ;blue to gray
StandardColorRGB2=255 255 255 ;green to white
StandardColorRGB3=255 255 255 ;cyan to white
StandardColorRGB7=0 0 0 ;light gray to black
StandardColorRGB12=0 0 0 ;bright red to black
```

```
StandardColorRGB15=0 0 0 ;bright white to  
black
```

3.4 Visual Basic Considerations

This section discusses the Visual Basic program requirements of an Instant Vinny application and the relationship of the components that link Instant Vinny and VB.

3.4.1 Visual Basic Requirements

An Instant Vinny application requires a Visual Basic project be created in the same manner as a standard Visual NPL application does. In addition, the module **INSTVNPL.BAS** must be added to that project.

This module includes the three basic procedures that are used by all Visual Basic controls for sending keystroke commands back to the NPL application. Those procedures are:

VnSendKeys

VnSendStr

VnSendSF

The **VnSendKeys** procedure is simple to use. Its purpose is to send either standard ASCII characters to the NPL application (including NPL virtual keys) or a single special function key.

The latter two procedures are specialized procedures. The **VnSendStr** procedure sends only standard keystrokes, while the **VnSendSF** procedure sends only special function keys.

Note **VnSendSF** is limited to sending a single special function key at a time. **VnSendStr** and **VnSendKeys** may send a string of characters limited in length only by the liberal character string restrictions in Visual Basic.

The Visual Basic project must contain all of the forms that you wish to display from your NPL program. In addition to **INSTVNPL.BAS** mentioned above, it must also contain

VNPLLINK.FRM, **VNPLUTIL.BAS**, and **VNPLDEV.BAS** as is required for all Visual NPL / VB projects.

In addition, **VNPLDEV.BAS** must be edited to include the object names of the forms you will be displaying, just as in Visual NPL.

Refer to Section 2.4 of your Visual NPL documentation for a discussion in how to add files to the VB projects and what must be modified in `VNPLDEV.BAS`.

3.4.2 Visual Basic Form Requirements

The Visual Basic forms used with Instant Vinny are fundamentally different from those used by Visual NPL.

First, the controls which you place on a form can only be 'clickable' in nature (i.e. push buttons and radio buttons.) You may also use the VB menu editor to create menu events that send keystrokes back to your NPL application.

Keyboard input controls like a TextBox will not work with Instant Vinny. In addition, mixed controls such as combo boxes are not recommended since these are not intended for use in the Instant Vinny environment.

It is important to remember that the Visual Basic form never retains focus for very long, as RTIWIN is predominantly obtaining focus in order to execute the NPL applications code.

Note To dress up a form, add 'display' controls such as text labels and picture boxes to the form. You may add as many of these controls as you wish.

The second difference between Instant Vinny and standard Visual NPL for is that all Instant Vinny forms require a special 'RTIWIN sync' control. By default this control is called **RTIWINFrame** and is just a standard text box control with a special name. The size and position of this control is what determines where the NPL window appears on the VB form.

The **Form_Load()** event of each VB form must contain a call to the Instant Vinny procedure **VnSetHost**. It is that call which tells RTIWIN to align itself with the special sync control placed on the VB form.

The NPL program controls the decision of which VB form is currently displayed by making `VnIVSelect` calls. It is not advisable to use VB commands to directly change forms or even

unload the current form. A better method is to send a keystroke back to the NPL application which will then execute a segment of code which will change the VB form to the one it wants.

It is a good idea to disable the closing of a VB form by modifying the formsQueryUnload event by adding the following code.

```
Private Sub Form_ QueryUnload(Cancel As  
Integer, UnloadMode As Integer)  
    If UnloadMode <> 1 Then  
        Cancel = 1  
        VnSendKeys NPL_Cancel  
    End If  
End Sub
```

The above code disables the event from automatically unloading the form by setting Cancel to 1 and sending the Cancel key to the NPL application. The NPL application will react to the cancel key by switching forms.

3.5 Mixing Instant Vinny with Visual NPL

An application may utilize both the Instant Vinny approach and Visual NPL development design. In fact, this is the recommended model for converting an existing NPL application to a Windows event driven application. The NPL application is first “Instant Vinnyized” and then some or all of its screens are converted to true Visual NPL forms.

Hint It is important to remember that when mixing the two, both cannot be active at the same time. You may have an Instant Vinny form visible (with RTIWIN visible and executing) or you may have one or more Visual NPL forms visible with RTIWIN hidden and “put to sleep”, but both may not be visible at the same time.

The easiest way to do this (and highly advised) is to make both part of the same VB project. Initialize Instant Vinny and then display either Visual NPL screens or Instant Vinny screens as your code executes.

You load an Instant Vinny form with the `IVSelect` procedure and you put Instant Vinny on hold by issuing `IVSelect` again without a form name. You may then launch a Visual NPL form using the methods normally used by Visual NPL programs. After returning from the Visual NPL portion of your application, Instant Vinny may be activated with another `IVSelect` call to the appropriate Instant Vinny form name.

Note It is permissible to have separate VB applications for both Visual NPL and Instant Vinny sections, but this method can be more difficult to implement and has few benefits.

3.6 Examples

Instant Vinny is distributed with an example application and fully commented source code that demonstrates how to begin and how to change VB forms. Also included are several sections showing how to use some Visual NPL procedures and functions to make your Instant Vinny forms more dynamic. An example of mixing both Instant Vinny and Visual NPL is also included.

The examples contain many VB forms that can be used as templates for just about any Instant Vinny conversion. Refer to the inline comments to see how and why things are done on the Visual Basic side of Instant Vinny.

C H A P T E R 4

NPL Reference



This chapter contains:

- Detailed descriptions of the NPL variables
- Definitions and detailed descriptions of each of the NPL subroutines and functions

4.1 NPL Variables

`VnIVSnapWindow`

This variable defaults to `VnTrue` which causes VB forms that are loaded by `VnIVSelect()` to appear in the same location that the previous VB form was positioned at. When set to `VnFalse`, VB forms appear in the position set during their design. It is advisable to leave this variable set to `VnTrue` as this provides a more logical and visually consistent display of forms. `VnIVSnapWindow` also controls the positioning of the first VB form based on the position of the VB form when it was last run. If `VnIVSnapWindow` is set to `VnTrue` then the Instant Vinny application will startup in the position that it was left off in. If false, it will default to the form's design position.

4.2 NPL Procedures

4.2.1 'VnIVCurrForm\$

Syntax FUNCTION'VnIVCurrForm\$

Parameters None

Description This function returns the name of the currently loaded VB form. This applies only to VB forms loaded byVnIVSelect , and not VB forms loaded using other Visual NPL commands.

Returns The name of the currently loaded Instant Vinny VB form

Example PROCEDURE 'Other
 DIM PrevForm\$128
 ;
 PrevForm\$= 'VnIVCurrForm\$
 'VnIVSelect("Other")
 ...
 'VnIVSelect(PrevForm\$)
 END PROCEDURE

See Also 'VnIVSelect

4.2.2 'VnIVDisableColor

Syntax PROCEDURE 'VnIVDisableColor

Parameters None

Description This procedure disables color support in RTIWIN. It may be useful in standardizing the display of applications converted to Instant Vinny mode.

Returns None

Examples 'VnIVDisableColor
'VnIVInit("Other")

4.2.3 'VnIVInit

Syntax PROCEDURE 'VnIVInit(
 /POINTER _AppName\$)

Parameters _AppName\$

Description This procedure initializes an Instant Vinny application. The parameter ***AppName\$*** is the name of the Visual Basic executable associated with the Instant Vinny application. If ***AppName\$*** is a blank string, Instant Vinny is initialized without doing a 'VnOpen. This should only be done if you are starting up Instant Vinny from a Visual NPL application (which will have already done a 'VnOpen).

Returns None

Example 'VnIVInit("MyApp ")

4.2.4 'VnIVSelect

Syntax PROCEDURE 'VnIVSelect(
 /POINTER
_FormName\$)

Parameters _FormName\$

Description This procedure loads a new VB form (**_FormName\$**) and positions it to the location of the existing VB form (if there is one). The previous VB form is then unloaded. Auto-positioning of the new VB form can be disabled by setting the variable `VnIVSnapWindow` to the value of `_VnFalse`. The NPL window (RTIWIN) is made visible if it isn't already. If the value of **_FormName\$** is a blank string, the existing form is unloaded and RTIWIN is made invisible. This is useful for switching to Visual NPL operation.

Returns None

Example 'VnIVSelect("Utility")

4.2.5 'VnIVShutdown

Syntax PROCEDURE 'VnIVShutdown/EXIT

Parameters None

Description This procedure shuts down an Instant Vinny application. It is called implicitly when the InstVnpl module is unloaded, but may be called explicitly if you wish to switch to Visual NPL mode and have a separate application preparing to execute.

Returns None

Examples 'VnIVShutdown

\$END

This page intentionally left blank

C H A P T E R 5

Visual Basic Reference



This chapter contains:

- Detailed descriptions of all Visual Basic constants
- Definitions and detailed descriptions of each of the VB subroutines

5.1 Constants

There are two sets of constants defined in the `instVnpl` module. One set for standard NPL keys, another for special function keys. The following sections identify each of the constants defined in this module.

5.1.1 Standard Keys (strings)

`NPL_Backspace`

`NPL_Return`

`NPL_Execute`

`NPL_Continue`

`NPL_Load`

`NPL_LineErase`

5.1.2 Special Function Keys (integers)

`NPL_Prev`

`NPL_Next`

`NPL_South`

`NPL_North`

`NPL_Erase`

`NPL_Delete`

`NPL_Insert`

`NPL_East`

`NPL_West`

`NPL_Recall`

`NPL_ShiftCancel`

NPL_ShiftPrev
NPL_ShiftNext
NPL_ShiftSouth
NPL_ShiftNorth
NPL_ShiftDelete
NPL_ShiftInsert
NPL_ShiftEast
NPL_ShiftWest
NPL_DTab
NPL_GL
NPL_ShiftGL
NPL_Tab
NPL_FN
NPL_ShiftTab
NPL_Underline
NPL_Cancel
NPL_Edit

5.2 Subroutines

5.2.1 VnGoNPL

Syntax Sub VnGoNPL()

Parameters None

Description This procedure is used to immediately return focus to the NPL window (RTIWIN). It is not required, as RTIWIN will automatically regain focus from VB in approximately one-tenth of a second after the mouse is released. Also procedure calls such as `VnSendKeys` implicitly return control to RTIWIN.

Returns None

Example **Private Sub cmdQuit_Click()**
 Unload Me
 VnGoNPL
 End Sub

5.2.2 VnSendKeys

Syntax Sub VnSendKeys(

Keys As Variant)

Parameters **Keys** Either a character string or numeric special function key

Description This procedure is a generalized form of VnSendStr and VnSendSF . If the parameter Keys is a string then it is passed to VnSendStr . Otherwise the parameter is converted to a one byte integer and passed to VnSendSF . Do not mix standard keystrokes and special function keys together with this procedure. Note the defined constants available for sending NPL virtual keys to the runtime.

Returns None

Examples Private Sub Form_Load()
 VnSendKeys NPL_Return
 End Sub

See Also VnSendStr

 VnSendSF

5.2.3 VnSendSF

Syntax	Sub VnSendSF FunctionKey%
Parameters	FunctionKey%
Description	This procedure sends a special function key to the NPL program. FunctionKey is converted to a single byte integer which corresponds to the hex codes defined for the special function NPL key.
Returns	None

Example **Private Sub cmdCancel_Click()**
 ' sends the cancel key and then SF key
 #5 to
 ' the NPL application
 VnSendSF NPL_Cancel
 VnSendSF 5

 End Sub

See Also VnSendStr
 VnSendKeys

5.2.4 VnSendStr

Syntax	Sub VnSendStr(KeyStrokes\$)
Parameters	KeyStrokes\$
Description	This procedure sends a character string representing standard NPL keystrokes to the NPL application. Do not use it to send special function keys. They are converted to a string representation which will produce illogical and erratic results.

Returns None

Example

```
Private Sub cmdYes_ Click()  
    ' sends a line erase key, the letter Y,  
    and  
    ' then the return key to the NPL  
    application.  
    ' Note that NPL_LineErase and  
    NPL_Return are  
    ' standard keys, not special function  
    keys  
    VnSendStr NPL_LineErase & "Y" &  
    NPL_Return  
End Sub
```

See Also VnSendKeys
VnSendSF

5.2.5 VnSetHost

Syntax	Sub VnSetHost <i>CtrlAs Control</i>
Parameters	<i>Ctrl</i> The name of the control used to sync position with RTIWIN
Remarks	This procedure is meant to be called from the Form_Load event procedure. It is required in every Instant Vinny form so that the NPL window will properly position itself on the VB form.
Returns	None

Example	<pre>Private Sub Form_Load() ' syncs the location of the RTIWIN window ' with the position of control RTIWINFrame VnSetHost RTIWINFrame End Sub</pre>
---------	---

C H A P T E R 6

Deployment



This chapter discusses how to create distribution diskettes for your Instant Vinny application. For a complete discussion about distributing Instant Vinny and Visual NPL applications, refer to Chapter 6 of the Visual NPL Developer's Guide.

6.1 Using the Visual Basic Setup Wizard

The following discusses the procedure for building distribution diskettes that include all the files needed to deploy an Instant Vinny application. With a few minor exceptions, this procedure is the same used to create distribution diskettes for a regular Visual NPL application. The Visual Basic Setup Wizard automatically includes all required Visual Basic files and libraries. The required NPL files and libraries are then included manually. Upon completion, a distribution set of all the Instant Vinny application and support files will be compiled.

1. In the program group for Visual Basic, select the Application Setup Wizard.
2. Enter the path to your Visual Basic Project(MBP) file or locate it using the browse feature.
3. Click on the **Next** button to bypass STEP 2.
4. In STEP 3 you have to select the method of distribution. In most cases distribution will be by floppy diskette.
5. Accept the default selections in STEP 4, 5, and 6.

6.1.1 Manually include the NPL files

The Application Setup Wizard analyzes the Visual Basic Project and automatically selects the files needed for the Visual Basic side. You will have to manually add the NPL related files to the Setup Wizard.

In STEP 7 you will tell the setup program which additional files to add to the setup. Using the **Add Files** feature, add:

All NPL libraries that are referenced including **VNPL.NPL**

All NPL diskimages that are referenced

All the Object (OBJ) files that are needed

VNPL16.DLL

Once you have added all the NPL files, click on the **Finish** key to finish creating the setup diskettes.

Note Visual Basic is not needed to run an Instant Vinny application, but a NPL Windows RunTime, Revision 4.22.21 or greater is needed.

Deploying Visual NPL files manually is discussed in Section 6.3 of the Visual NPL Developer's Guide.

When run, the Setup routine made using the Visual Basic Application Setup Wizard will create a Program Group and a Shortcut to the executable file. Both the Program Group and the Shortcut should be deleted. They are not needed, the Visual Basic .EXE file is launched by the `onIVInit (<file name>)` procedure in the NPL program and cannot be launched directly.

The Application Setup Wizard supplied with Visual Basic has no provision to prevent creating the program group or the Shortcut. Third party Setup programs are available that may optionally prevent the creation of a Program Group and Shortcut.

