

Visual Basic™ Extensions for Modular Windows



Microsoft® MODULAR WINDOWS™ SOFTWARE DEVELOPMENT KIT

Visual Basic™ Extensions for Modular Windows

Microsoft® Modular Windows™ Software Development Kit

Version 2.0M

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

©1992 Microsoft Corporation. All rights reserved.

Microsoft, MS, MS-DOS, and the Microsoft logo are registered trademarks, and Windows and Visual Basic are trademarks of Microsoft Corporation in the USA and other countries.

Intel is a registered trademark and Indeo is a trademark of Intel Corporation.
Tandy is a registered trademark and VIS is a trademark of Tandy Corporation.
TrueType is a registered trademark of Apple Computer, Inc.

Other names are the trademarks and registered trademarks of their respective owners.

Amadeus courtesy of The Saul Zaentz Company. Copyright © 1983 by The Saul Zaentz Company.
All rights reserved.

At Play in the Field of the Lords courtesy of The Saul Zaentz Company. Copyright © 1991 by
The Saul Zaentz Company. All rights reserved.

CNN News courtesy of Cable News Network, Inc. Copyright © 1992 by Cable News Network, Inc.
All rights reserved.

Dollars and Sense courtesy of Headline News Network. Copyright © 1992 by Cable News Network,
Inc. All rights reserved.

King5 News, Rescue Northwest, Almost Live, Evening Magazine courtesy of King5 TV.
Copyright © 1992 by King5 Television. All rights reserved.

Matlock, Fun in Acapulco, Andy Griffith Show, and Nails courtesy of Viacom Enterprises. Copyright
© 1992 by Viacom Enterprises, a division of Viacom International Inc. All rights reserved.

MTV ID courtesy of MTV Networks. Copyright © 1992 by MTV Networks, a division of Viacom
International Inc. All rights reserved.

NFL Gameday promotion courtesy of ESPN. Copyright © 1992 by ESPN. All rights reserved.

Contents

Chapter 1 Introduction	1-1
Document Conventions	1-2
What is Visual Basic Extensions for Modular Windows?	1-3
Custom Controls for Modular Windows	1-3
The Run-Time Environment	1-3
The Design Environment	1-4
The Modular Windows API Header File	1-4
Sample Applications	1-4
The Setup Program	1-4
Installing Visual Basic Extensions for Modular Windows	1-4
Before You Run Setup	1-5
Running the Setup Program	1-5
After Installation	1-6
File Locations	1-6
Changes to System Files	1-7
Online Help	1-8
Redistributable Run-Time Modules	1-8
Chapter 2 Changes and New Features	2-1
Changes and New Features for the Design Environment	2-1
The Design-Time Executable File	2-2
The Run-Time Executable File	2-2
Menu Changes in Visual Basic Extensions	2-2
The Convert Project Command	2-2
The New MDI Form Command	2-3
The Menu Command	2-3
The Toolbox in Visual Basic Extensions	2-4
Converting Visual Basic 2.0 Applications to Modular Windows	2-4
Control Mapping	2-5
Property Mapping	2-6
The Run and Make EXE Commands	2-6

Changes in the Run-Time Environment	2-7
Error Handling	2-7
Unsupported API Calls	2-7
Language Features	2-8
Chapter 3 Custom Control Reference	3-1
About the Custom Controls	3-1
What Is a Custom Control?	3-1
Loading Custom Controls	3-2
The Toolbox	3-2
Distributing Applications that Use Custom Controls	3-4
Setting the Value of Control Parameters	3-4
Object Type	3-5
Distribution Note	3-5
Properties, Events, and Methods	3-5
Check Box	3-6
Edit Box	3-8
Group box	3-10
Keyboard	3-12
List Box	3-14
Option Button	3-16
Push Button	3-18
Scroll Bar (Horizontal)	3-20
Scroll Bar (Vertical)	3-22
Scroll Pad	3-24
Show Box	3-26
Sticky Button	3-28
Properties Reference	3-30
AlignOffset	3-30
AlignStyle	3-30
BackColor	3-31
BitmapDown, BitmapFace, and BitmapUp	3-32
BitmapThumb	3-33
ChannelFace, ChannelFrame, ChannelHight, ChannelShadow, ChannelTextCol	3-34
DefaultKey	3-35
DefaultLength	3-35
DefaultText	3-36
DisplayOnly	3-36
FaceColor, FrameColor, HightColor, ShadowColor, and TextColor	3-37

Font	3-38
FontSelDelta	3-38
KbdHwnd	3-39
KeyboardXPos	3-39
KeyboardYPos	3-40
Name	3-40
NoEdge	3-40
NoFrame	3-41
PadStyle	3-41
Prompt	3-42
RectStyle	3-42
Reset	3-43
SelectString	3-43
Separation	3-43
Slider	3-44
SmallChange	3-44
Sorted	3-44
Text	3-45
TextHAlign	3-45
TextVAlign	3-46
ThumbFace, ThumbFrame, ThumbHight, ThumbShadow, ThumbTextCol	3-46
ThumbInside	3-47
TransColorDown, TransColorFace, TransColorUp	3-48
TransColorThumb	3-49
TransModeDown, TransModeFace, and TransModeUp	3-50
TransModeThumb	3-50
TVdown, TVleft, TVright, TVup	3-51
Wakeup	3-52
Chapter 4 Online TV Program Guide	4-1
Program Features	4-2
Hardware and Software Requirements	4-3
Installation and Operation	4-4
Program Modification	4-8
Redistribution Rights	4-11

Introduction

Visual Basic™ Extensions for Microsoft® Modular Windows™ is a powerful development tool you can use to create applications for Modular Windows. Before you can install and use Visual Basic Extensions, you must install Visual Basic for Windows. It is recommended that you also install Modular Windows and the Modular Windows SDK prior to installing Visual Basic Extensions. This will allow you to test your applications under the Modular Windows Operating System.

You can use Visual Basic Extensions to port existing Visual Basic applications to Modular Windows, or you can create new applications written especially for Modular Windows. Visual Basic Extensions and Visual Basic 2.0 have similar interfaces, so you can transfer most of your Visual Basic 2.0 experience directly into working with Visual Basic Extensions.

This manual provides the following information:

- A list of differences between Visual Basic 2.0 and Visual Basic Extensions
- Descriptions of custom controls designed for Visual Basic Extensions
- Instructions for installing Visual Basic Extensions and the Modular Windows custom controls
- An alphabetic reference of the Modular Windows custom controls
- Information about the TV Program Guide, a sample application written with Visual Basic

Your package might include a readme file that contains helpful information you can use in addition to this manual.

Document Conventions

This chapter uses the same typographic conventions as the *Microsoft Visual Basic Custom Control Reference*.

Convention	Meaning
Bold text	Words in bold with initial letter capitalized indicate language-specific keywords with special meaning to Visual Basic; for example, GetData .
Initial Capitals	Names of properties, events, and special objects appear with initial letter capitalized; for example, BackColor.
<i>Italic text</i>	In syntax, italic letters indicate place holders for information you supply; for example, <i>setting%</i> .
[]	In syntax, items inside square brackets are optional.
{ }	In syntax, braces and a vertical bar indicate a mandatory choice between two or more items. You must choose one of the items unless all of the items also are enclosed in square brackets; for example, {whileuntil}.
monospace	Indicates sample code.
ALL CAPITALS	Words in all capital letters indicate filenames or constants; for example, CONST2.TXT. Acronyms are usually spelled out the first time they are used; for example, “Object Linking and Embedding (OLE).”
SMALL CAPITALS	A plus (+) sign between key names indicates a combination of keys. For example, ALT+F1 means to hold down the ALT key while pressing the F1 key.

What is Visual Basic Extensions for Modular Windows?

Visual Basic Extensions is a powerful tool you can use to develop applications for Modular Windows. Before you can use Visual Basic Extensions, you must have Visual Basic for Windows installed. It is recommended that you also have Modular Windows and the Modular Windows SDK installed.

Visual Basic Extensions include the following:

- Visual Basic (VB) custom controls for Modular Windows
- Visual Basic run time for Modular Windows (VBR200MW.DLL)
- Visual Basic design environment for Modular Windows
- The Modular Windows API Header File
- Sample files
- Visual Basic 2.0 for Modular Windows Setup program

Custom Controls for Modular Windows

A custom control is an extension to the Visual Basic Toolbox. The custom controls included with Visual Basic Extensions for Modular Windows are specially designed to be used on television displays.

Once you install the Modular Windows custom controls, you can use them just as you would any of Visual Basic's built-in controls. When you add a custom control to a project in Visual Basic Extensions, it becomes part of the Visual Basic development environment and provides new functionality to your applications.

The Visual Basic 2.0 for Modular Windows Setup program automatically installs the custom controls for you. See "Installing Visual Basic Extensions," later in this chapter, for more information.

The Run-Time Environment

The run-time file used by Visual Basic Extensions is VBR200MW.DLL. This file is automatically installed when you run the Setup program. See "Installing Visual Basic Extensions," later in this chapter, for more information about the Setup program.

The Design Environment

The design environment in Visual Basic Extensions is similar to that of Visual Basic 2.0. Some of the controls are different, and a few menu items are different. The basic procedures you use to create an application are the same. However, some of the Visual Basic 2.0 statements and methods are not supported in Visual Basic Extensions. For a list of these statements and methods, see Chapter 2, “Changes and New Features.”

The Modular Windows API Header File

Visual Basic Extensions includes a file, `MODW_API.TXT`, that includes all the APIs supported by Modular Windows. When you create applications for Modular Windows, be sure to use this file for external declarations if your application uses any direct Windows calls.

You should use `MODW_API.TXT` instead of `WIN30API.TXT` and `WIN31EXT.TXT`, the two API files provided with Visual Basic 2.0. `MODW_API.TXT` is not an include file. Copy the information you need from the file into your application’s Global module.

Sample Applications

Visual Basic Extensions includes several sample applications. Chapter 4, “Online TV Program Guide,” provides detailed information about one of the sample applications.

The Setup Program

The Setup program installs the files you need to run Visual Basic Extensions for Modular Windows. See the following section, “Installing Visual Basic Extensions for Modular Windows,” for the complete installation procedure.

Installing Visual Basic Extensions for Modular Windows

The Setup program for Visual Basic Extensions for Modular Windows lets you choose which components of the product you want to install. If you install all components available, you must also have several support files on your system. These files include:

- Run-time support files from Microsoft Video for Windows
- Dynamic-link libraries (DLLs) from the Modular Windows SDK
- Files from Microsoft Visual Basic for Windows

Before You Run Setup

You must install Visual Basic on your system before you install Visual Basic Extensions for Modular Windows. The Setup program will not run if you have not installed this product. The files for each version of Visual Basic will remain separate from each other so that you can develop applications for use on PCs or applications for use on TV-based players. It is recommended that you also install Modular Windows and the Modular Windows SDK before installing Visual Basic Extensions for Modular Windows.

The Modular Windows SDK CD-ROM includes three setup programs: one to install the SDK itself, one to install Visual Basic Extensions for Modular Windows, and another to install the run-time support files from Video for Windows. If you have already installed Video for Windows, you will not need to use the third Setup program mentioned. If you have not installed Video for Windows, the Visual Basic Extensions for Modular Windows Setup program will create an icon you can use to install the required support files from the Modular Windows SDK CD-ROM.

For information about the Modular Windows SDK Setup program, see the Modular Windows SDK documentation.

The Visual Basic Extensions for Modular Windows Setup program is in the \VB directory on the CD. The setup program for Video for Windows is in the \WINVIDEO directory.

Running the Setup Program

The Visual Basic Extensions for Modular Windows Setup program installs all required files and creates a new Program Manager group called “VB for Modular Windows.”

► **To install Visual Basic Extensions for Modular Windows:**

1. Install Visual Basic for Windows on your system.
2. If you want to test the application under Modular Windows on your development system, install Modular Windows and the Modular Windows SDK on your system.

This level of testing is highly recommended.

3. Insert the Modular Windows SDK CD into the CD-ROM drive and change to the drive.
4. Enter the following commands to change to the VB directory and start the Setup program:

```
cd vb
vbsetup.exe
```
5. Follow the instructions the Setup program displays on your screen.

After Installation

The Visual Basic Extensions for Modular Windows Setup program copies files to your hard disk and updates your system files so that Visual Basic Extensions for Modular Windows is properly configured. This section tells you the default locations for the new files, and explains the changes made to system files.

File Locations

The Visual Basic Extensions for Modular Windows Setup program copies all the files required for the components you selected during installation. The following table identifies the default location for files used by Visual Basic Extensions for Modular Windows. You can change some of these names by entering your preferred location during installation. The Setup program will prompt you for destinations.

Location	Contents
WINDOWS	VB.INI
MWDIST	MCIOL.DLL
	MSVIDEO.DLL
	INDEO.DRV
	MCIAVI.DRV
	MSVIDC.DRV
	MPLAYER.EXE
	MPLAYER.REG
	MPLAYER.HLP
	PROFDISP.EXE
	HC.DLL
	TVUI.DLL

Continued

Location	Contents
MWDIST (continued)	DISPDIB.DLL VBR200MW.DLL TVBTN.VBX TVEDT.VBX TVKBD.VBX TVLIST.VBX TVPAD.VBX TVSCROLL.VBX TVSTA.VBX
VB (or your Visual Basic directory)	VBMW.EXE MODW_API.TXT
MODWIN\VBSAMPLETICTAC	Tic-tac-toe sample
MODWIN\VBSAMPLE\CDPLAYER	CD Player sample
MODWIN\VBSAMPLE\TVGUIDE	TV Program Guide sample

Changes to System Files

The Visual Basic Extensions for Modular Windows Setup program adds a [TVUI] section to your SYSTEM.INI file. This section includes one command:

```
SetSysDefColors=Yes
```

The value specified in this command determines what happens to your desktop background when you load any of the .VBX files from Visual Basic Extensions for Modular Windows. If you use the default value (YES), your desktop background is set to gray when you load the .VBX files. When you remove the Modular Windows custom controls, or when you exit Visual Basic Extensions for Modular Windows, the desktop background color is restored to whatever it was before you loaded the custom controls.

If you don't want the desktop background to change when you load the custom controls, set this value to NO.

Online Help

Online Help is available from Visual Basic for Modular Windows. The Help information is the same that is available in your existing Visual Basic installation. However, there are some differences between Visual Basic and Visual Basic for Modular Windows that are not reflected in online Help. See Chapter 2, “Changes and New Features,” for information about these differences.

Redistributable Run-Time Modules

You may redistribute the following run-time modules from the Microsoft Modular Windows SDK when these files are required by your application:

- DISPDIB.DLL
- HC.DLL
- HCSTUB.DLL
- HCVIS.DLL
- LRHELV.FON
- MCMAN.EXE
- TVBTN.VBX
- TVEDT.VBX
- TVKBD.VBX
- TVLIST.VBX
- TVPAD.VBX
- TVSCROLL.VBX
- TVSTA.VBX
- TVUI.DLL
- TVVGA.FON
- VBR200MW.DLL

The GDI.EXE, USER.EXE, and VBMW.EXE files are **not** redistributable. For more information about redistributable run-time modules, please read Section 5 in the separate Microsoft Modular Windows SDK license agreement.

The multimedia video and audio materials included in the Modular Windows SDK are all copyrighted, and may not be reproduced, distributed, or publicly performed. For a listing of the copyright owners to contact regarding permission and use rights for these materials, see the copyright page of this manual. Note that the Modular Windows SDK does not contain any sample video or audio materials that you may redistribute as mentioned in Section 3 of the Microsoft Modular Windows SDK license agreement.

Changes and New Features

You will find many similarities between Visual Basic 2.0 and Visual Basic Extensions. You will also notice some differences, mostly due to the differences between target environments. Some of these differences affect you at design time, others affect you at run time, and some affect design time as well as run time. This chapter describes the differences between Visual Basic Extensions and Visual Basic 2.0.

Visual Basic 2.0 applications are designed to run on a personal computer, while applications from Visual Basic Extensions are designed to run on a TV-based player. Each of these target environments presents different development challenges. The standard controls and custom controls available to you in Visual Basic Extensions are designed to help you quickly and easily develop effective, interesting applications for TV-based players.

Many of the commands from Visual Basic 2.0 are supported directly in Visual Basic Extensions, although some are not. This chapter includes information about commands that function differently, as well as information about commands from Visual Basic 2.0 that are not supported in Visual Basic Extensions.

Changes and New Features for the Design Environment

Some changes that affect the design environment are apparent when you look at the menus in Visual Basic Extensions. Other changes are less obvious. The changes and new features in the design environment include the following:

- A different Visual Basic design-time executable file (VBMW.EXE)
- A different, smaller run-time file (VBR200MW.DLL)
- Disabled menu commands
- New menu commands
- Different controls from Visual Basic 2.0
- Conversion support for existing Visual Basic applications

The Design-Time Executable File

The executable file for Visual Basic Extensions is called VBMW.EXE. When you double-click the VB for Modular Windows icon this file is executed. The changes and new features that affect the design-time executable are described in “Changes and New Features in the Design Environment,” later in this chapter.

While you are using VBMW.EXE, only the controls that are supported by Modular Windows will be available for you to add to an application. The procedure for selecting and adding controls is the same as that for Visual Basic 2.0.

You can convert Visual Basic 1.0 or 2.0 applications to run under Modular Windows.. See “Converting Visual Basic Applications to Modular Windows,” later in this chapter, for more information.

The Run-Time Executable File

VBR200MW.DLL is the run-time executable version of Visual Basic Extensions. When you use Visual Basic Extensions to compile an executable file, Visual Basic will automatically use VBR200MW.DLL.

You can run Modular Windows applications on a regular PC as long as you install the following files on the system:

- VBR200MW.DLL
- TVUI.DLL
- HC.DLL
- The .VBX files that support any Modular Windows custom controls in your application

Menu Changes in Visual Basic Extensions

Visual Basic Extensions includes a new menu command that helps you convert existing Visual Basic applications into applications for Modular Windows. This section describes this new command and also tells you about other menu changes in Visual Basic Extensions.

The Convert Project Command

The File menu in Visual Basic Extensions includes a new command, Convert to Modular Windows. When you choose this command, Visual Basic saves the current project in text format and then reloads the project. As the project is reloaded, Visual Basic maps Visual Basic controls to their equivalents in Visual Basic Extensions.

The mapping is controlled by the settings in the [ControlMap] section of VB.INI. For more information about this section, see “Control Mapping,” later in this chapter.

Visual Basic creates log files during the conversion and form view processes. If errors occur during project conversion, Visual Basic records the errors in the log files. Log file names correspond to form names. For example, file ABC.LOG records errors that occurred while converting form ABC.FRM.

For more information about converting applications, see “Converting Visual Basic Applications to Modular Windows,” later in this chapter.

► **To use the Convert Project command:**

1. From the File menu, choose Open Project to open an existing Visual Basic project.

2. From the File menu, choose Convert Project.

If you have not previously converted this project to Modular Windows, Visual Basic displays the following message: “Convert all controls in this project to Modular Windows controls?”

3. To continue, choose the OK button.

Visual Basic displays the Save Converted File dialog box.

4. In the dialog box, select the drive and directory where you want to save the files for the Modular Windows project. Also enter a name for the Modular Windows project.

For more information about the tools available in Visual Basic Extensions, see “The Toolbox in Visual Basic Extensions,” later in this chapter.

The New MDI Form Command

The New MDI Form command from Visual Basic 2.0 is not supported in Visual Basic Extensions.

The Menu Command

You can use the Menu command on the Window menu to examine existing menus, but not to create new menus. The Menu command is only enabled when the current form has a menu.

The Toolbox in Visual Basic Extensions

Some of the Visual Basic 2.0 tools are directly supported in Visual Basic Extensions. Others have been replaced by new Modular Windows controls, and some are not supported. This is because Visual Basic Extensions does not support controls that are not compatible with the TV-based player environment.

The following table identifies the status of Visual Basic 2.0 tools in Visual Basic Extensions:

Tool	Status
Pointer	Supported
Picture	Supported
Label	Supported
Text Edit	Replaced by TV Edit Box
Frame	Replaced by TV Group Box
Push Button	Replaced by TV Push Button
Check Box	Replaced by TV Check Box
Radio Button	Replaced by TV Option Button
Combo List Box	Replaced by TV List Box
List Box	Replaced by TV List Box
Horizontal Scroll Bar	Replaced by TV Horizontal Scroll Bar
Vertical Scroll Bar	Replaced by TV Vertical Scroll Bar
Timer	Supported
Drive List Box	Not supported
Directory List Box	Not supported
File List Box	Not supported
Shape Tool	Supported
Line Tool	Supported
Image Tool	Supported

Converting Visual Basic 2.0 Applications to Modular Windows

You can load existing Visual Basic projects and forms into Visual Basic Extensions. When you save a project using Visual Basic Extensions, the project is saved as a Modular Windows project.

Some controls from your Visual Basic 2.0 applications will map automatically to corresponding controls supported by Visual Basic Extensions. If your project uses controls that cannot be mapped to Modular Windows controls, Visual Basic displays a message.

Control Mapping

Some controls from Visual Basic 2.0 are mapped to Modular Windows controls automatically when you load your Visual Basic 1.0 or 2.0 application into Visual Basic Extensions. The Visual Basic Extensions Setup program adds a section to your VB.INI file that identifies these controls and their Modular Windows replacements.

It is recommended that you not change the entries made by the Setup program. You can add entries to VB.INI to override certain default actions taken by Visual Basic Extensions when loading applications.

The [ControlMap] Section In VB.INI

Visual Basic Extensions automatically adds the [ControlMap] section to your VB.INI file and enters the names of Visual Basic 2.0 controls that can be mapped automatically to Modular Windows controls. If you create custom controls for Modular Windows, you can add mapping entries for those commands.

The syntax for the [ControlMap] section is as follows:

```
OldControlName = VBXFilename!NewControlName
```

The .VBX filename is optional. You can omit the filename to map to an internal Visual Basic control. If you specify a file, specify the full path to the file, including the drive letter.

If a control is unsupported and has no mapping equivalent in Visual Basic Extensions, the right side of the entry is blank, as in the following example:

```
OldControlName =
```

If the project you are converting contains controls that are not specified in VB.INI, Visual Basic assumes the controls don't need to be changed for Modular Windows.

For a complete list of fully supported, non-supported, and mapped controls, see "The Toolbox in Visual Basic Extensions," earlier in this chapter.

Overriding Defaults in VB.INI

By default, Setup enters the names of all Visual Basic 2.0 controls that have been removed from the Toolbox in Visual Basic Extensions. Except for mapping these tools to Modular Windows tools, they are not supported in the Visual Basic Extensions design environment. If no mapping equivalent was found for a control, it will appear in the form. You should replace these controls with one or more Modular Windows controls.

To override this default, you can include the following entry in the [ControlMap] section:

```
ShowOldControls=1
```

If you change the default behavior, controls that are not supported in Visual Basic Extensions will appear in the Toolbox. However, your project will not compile correctly if you include any of the unsupported commands in a form.

Also by default, Visual Basic Extensions automatically loads the .VBX files specified in the [ControlMap] section. To override this default, include the following entry in the [ControlMap] section:

```
AlwaysLoadNewControls=0
```

Property Mapping

When you map controls from Visual Basic 2.0 into Visual Basic Extensions, most of the control properties will also map automatically. Property mapping requires identical property names on both controls. If the properties have different names, Visual Basic will not map them. In these cases, you must manually assign values to the properties of the Modular Windows controls.

The Run and Make EXE Commands

If the current application includes controls that are not supported by Visual Basic Extensions, using either the Run command or the Make EXE command on the Run menu will cause an error. Visual Basic will display a message indicating which control in the current application is not supported.

To correct this problem, replace the unsupported control with a valid Modular Windows control, then try the command again.

Changes in the Run-Time Environment

Due to the differences between Windows 3.1 and Modular Windows, some Visual Basic 2.0 features are not supported in Visual Basic Extensions. These features include the following:

- Menus
- The system menu
- Sizing borders
- Minimize and maximize buttons
- Non-client scroll bars
- Controls based on the `BUTTON`, `COMBOBOX`, `EDIT`, `LISTBOX`, `SCROLLBAR`, and `STATIC` control classes
- MDI
- Common dialog boxes
- DDE
- Printing
- TrueType® fonts
- MAPI

Error Handling

You should carefully test and debug your application, eliminating as many error conditions as possible. Create code that is as robust as possible, anticipating user actions as much as you can.

On a TV-based player, Modular Windows provides only a general-purpose error message. Modular Windows does not provide specific error messages for all possible error conditions encountered.

Unsupported API Calls

API calls that are not supported in Modular Windows are not supported in Visual Basic Extensions. Calls that are supported with modifications in Modular Windows are similarly supported in Visual Basic Extensions.

For a complete list of unsupported and changed APIs, see the *Modular Windows Programmer's Reference* in the Modular Windows SDK.

Language Features

The Printer object from Visual Basic is not supported in Visual Basic Extensions.

The following methods are not supported:

- PrintForm
- NewPage
- EndDoc
- SavePicture
- LinkSend
- LinkPoke
- LinkRequest
- LinkExecute

MDI is not supported.

The following window styles are not supported:

- ControlBox
- MaxButton
- MinButton
- BorderStyle
- WindowState

Custom Control Reference

This chapter describes the custom controls available in Microsoft Visual Basic Extensions for Modular Windows. Use this chapter as a supplement to the *Microsoft Visual Basic Custom Control Reference*. You can also get more information from Visual Basic 2.0 online Help, which is directly accessible in Visual Basic Extensions.

Each control is described individually. For each control, the following information is given:

- A description
- Visual Basic icon
- A picture of the control
- The name of the .VBX file that supports the control
- The object type of the control
- When appropriate, remarks and notes to help you use the control

About the Custom Controls

Visual Basic Extensions includes custom controls designed specifically for use on TV-based players. These controls accommodate the differences between televisions and computer monitors, and the differences between Microsoft Windows 3.1 and Microsoft Modular Windows.

What Is a Custom Control?

A custom control is an extension to the Visual Basic Toolbox. You use custom controls just as you would any of the built-in controls for Visual Basic. When you add a custom control to a project, it becomes part of the Visual Basic development environment and provides new functionality to your applications.

Loading Custom Controls

A custom control is a *dynamic-link library* (DLL) that contains all the information Visual Basic needs to provide one or more new types of controls.

► **To load a custom control file:**

1. From the File menu, choose Add File.
2. Visual Basic displays the Add File dialog box with three proposed file extensions: *.FRM, *.BAS, and *.VBX. By convention, custom-control files have a .VBX extension.
3. Type or select the name of the custom control file you want to open. The .VBX files included with Modular Windows are located in your \MWDISTdirectory.
4. As soon as you load the file, the following happens:
 - The custom-control file appears in the Project window.
 - All the controls provided by this file are added to the Toolbox, which is extended to accommodate the new controls.
 - Your system colors will change—all white backgrounds become gray. You can change this default behavior. See “Overriding Defaults in VB.INI,” in Chapter 2, “Changes and New Features,” for more information.

You can now use the new controls just as if they were standard controls. You can add them to any of the forms in your project. Each custom control has its own set of properties and events, which you can see by using the Properties bar and the Code window. These properties are described at the end of this chapter.


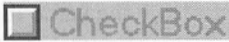



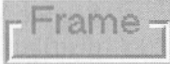



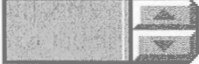

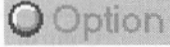



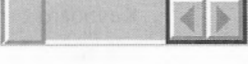

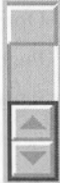





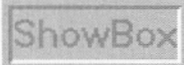
Note You can use custom controls with Visual Basic, but you cannot use Visual Basic to create custom controls. To create custom-control files yourself, see the *Microsoft Visual Basic Control Development Guide*.

The Toolbox

When you use Visual Basic Extensions, the only tools available in the Toolbox are those supported by Modular Windows.

To add the Modular Windows controls to the Toolbox, use the Add File command on the File menu. This is the same procedure you use in Visual Basic 2.0 to add custom controls to the Toolbox. To insert a custom control into your Visual Basic form, double-click the appropriate icon in the Toolbox.

The following table lists the name, icon, and image for each of the Visual Basic custom controls for Modular Windows.

Icon	Control Name	Control Image
	Check box	 CheckBox
	Edit box	 EditBox
	Frame box	 Frame
	Keyboard	
	List box	
	Option button	 Option
	Push button	
	Scroll bar, horizontal	
	Scroll bar, vertical	
	Scroll pad	
	Sticky button	
	Show box	

Distributing Applications that Use Custom Controls

You can use custom controls in applications that you distribute. A custom-control file (.VBX file) is a DLL that is accessed by Visual Basic. If your application uses custom controls, the .VBX file for those controls must be on the system path, or in the same directory as the application executable (.EXE) file. To distribute custom controls with your application, make sure your installation procedure copies all required .VBX files into the user's \WINDOWS\SYSTEM directory. If the application will be used on TV-based players, copy the required .VBX files into a CD directory that is on the path.

See the Visual Basic documentation for full instructions on creating, running, and distributing executable files.

The following table lists the .VBX files included with Modular Windows. It also identifies any other files required by the custom controls. When you distribute an application that uses any of these controls, be sure to include the appropriate .VBX file and any other required files with your application. For information about writing an installation program for your application, see the *Microsoft Visual Basic Programmer's Guide*.

Control	.VBX File
Check box	TVBTN.VBX
Edit box	TVEDT.VBX
Group box	TVBTN.VBX
Keyboard	TVKBD.VBX
List box	TVLIST.VBX
Option button	TVBTN.VBX
Push button	TVBTN.VBX
Scroll bar, horizontal	TVSCROLL.VBX
Scroll bar, vertical	TVSCROLL.VBX
Scroll pad	TVSCROLL.VBX
Show box	TVSTA.VBX
Sticky button	TVBTN.VBX

Setting the Value of Control Parameters

Some edit-control parameters can be set either with Visual Basic properties or through Windows messages. In these cases, use the Visual Basic properties to set the parameter values. Don't use the equivalent Windows message. If messages are used to change the parameters, the corresponding Visual Basic properties might not be updated correctly.

Object Type

A control's *object type* is used with the **TypeOf** keyword in an **If...Then...Else** statement. This is useful for creating a variable of a particular object type, or determining the type of control that is passed as an argument to an event (for example, the *Source* argument of the DragDrop event). For more information on using the *object type* parameter, search Visual Basic's online Help for the **If** keyword.

The following table identifies the object type for each custom control in Visual Basic Extensions.

Control Name	Object Type
Check box	MWCheckBox
Edit box	MWEdit
Group box	MWGroupBox
Pop-up keyboard	MWKeyboard
List box	MWList
Option button	MWOption
Push button	MWButton
Horizontal scroll bar	MWHScroll
Vertical scroll bar	MWVScroll
Scroll pad	MWPad
Show box	MWShowBox
Sticky button	MWSticky

Distribution Note

If your application uses a Modular Windows custom control, your Setup program should install the appropriate .VBX file in the user's Microsoft Windows \SYSTEM directory. The documentation for each control identifies the required .VBX file. The Visual Basic Setup Kit included with Visual Basic 2.0 provides tools you can use to write setup programs that install your applications correctly.

Properties, Events, and Methods

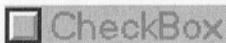
The documentation for each control includes a table of the properties, events, and methods supported by the control. Properties marked with an asterisk (*) are further documented later in this chapter. See the *Microsoft Visual Basic Custom Control Reference* or online Help for documentation of all other properties, events, and methods.

Check Box

The check box displays an option that can be turned on or off. The following illustration shows the Toolbar icon and control image for the Modular Windows check box:



Icon



Control

Filename TVBTN.VBX

Object Type MWCheckBox

Remarks The Value property for this control is compatible with the check box in Visual Basic 2.0. You can set the Value property to 0 or 1.

Properties, Events, and Methods

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Category	Items		
Properties	About	AlignOffset*	AlignStyle*
	BackColor*	BitmapDown*	BitmapFace*
	BitmapUp*	Caption	DragIcon
	DragMode	Enabled	FaceColor*
	Font*	FontBold	FontItalic
	FontName	FontSize	FontStrikethru
	FontUnderline	FrameColor*	Height
	hWnd	HightColor*	Index
	Left	MousePointer	Name
	Parent	Separation*	ShadowColor*
	TabIndex	TabStop	Tag
	TextColor*	Top	TransColorDown*
	TransColorFace*	TransColorUp*	TransModeDown*
	TransModeFace*	TransModeUp*	TVdown*
	TVleft*	TVright*	TVup*
	Value	Visible	Width
	Events	Click	DragDrop
GotFocus		KeyDown	KeyPress
KeyUp		LostFocus	
Methods	Drag	Move	Refresh
	SetFocus		

Edit Box

The Modular Windows edit-box control can be used on devices that don't have keyboards. To support such devices, a popup keyboard appears on the screen when a user chooses the edit-box control. The popup keyboard accepts input from a hand controller.

The following illustration shows the Toolbar icon and control image for the Modular Windows edit box:



Icon



Control

Filename TVEDT.VBX

Object Type MWEdit

Remarks Make sure this control and the popup keyboard will both be fully visible when the keyboard is displayed. To do this, put this control in a location that is not covered when the keyboard is in its default location. Or, you can set the location of the keyboard so it doesn't cover the edit box.

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

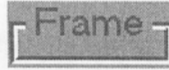
Group	Items		
Properties	About	DefaultKey*	DefaultLength*
	DefaultText*	DragIcon	DragMode
	Enabled	FaceColor*	Font*
	FontBold	FontItalic	FontName
	FontSize	FontStrikeThru	FontUnderline
	FrameColor*	Height	HighltColor*
	hWnd	Index	KeyboardXPos*
	KeyboardYPos*	Left	Name
	Parent	ShadowColor*	TabIndex
	TabStop	Tag	Text*
	TextColor*	Top	TVdown*
	TVleft*	TVright*	TVup*
	Visible	Width	
	Events	Change	DragDrop
GotFocus		LostFocus	SoftChar
Methods	Drag	Move	Refresh
	SetFocus		

Group box

A group box provides a graphical or functional grouping of controls. The following illustration shows the Toolbar icon and control image for the Modular Windows group box:



Icon



Control

Filename

TVBTN.VBX

Object Type

MWGroup

Remarks

The group box, or frame, is useful for defining a group of controls that are used together.

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Category	Items		
Properties	About	AlignOffset*	AlignStyle*
	BitmapFace*	Caption	DragIcon
	DragMode	Enabled	FaceColor*
	Font*	FontBold	FontItalic
	FontName	FontSize	FontStrikethru
	FontUnderline	FrameColor*	Height
	hWnd	HightColor*	Index
	Left	MousePointer	Name
	Parent	Separation*	ShadowColor*
	TabIndex	TabStop	Tag
	TextColor*	Top	TransColorFace*
	TransModeFace*	TVdown*	TVleft*
	TVright*	TVup*	Visible
	Width		
	Events	Click	DragDrop
GotFocus		KeyDown	KeyPress
KeyUp		LostFocus	
Methods	Drag	Move	Refresh
	SetFocus		

Keyboard

The TV keyboard control displays on the television a keyboard that responds to the Modular Windows hand control. The keyboard icon appears only at design time. The keyboard controls appears during run time when the application assigns any value to the Wakeup property. The following illustration shows the Toolbar icon and design-time control image for the Modular Windows popup keyboard:



Icon



Control

Filename TVKBD.VBX

Object Type MWKbd

Remarks This control is invisible to the user at run time until a value is assigned to the keyboard's Wakeup property.

Note The location of the icon within a form does not determine the location of the keyboard. Use the KeyboardXPos and KeyboardYPos properties to set the location of the keyboard.

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

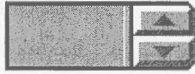
Group	Items		
Properties	About	DefaultKey*	DefaultLength*
	DefaultText*	FaceColor*	FrameColor*
	HighltColor*	hWnd	Index
	KbdHwnd*	KeyboardXPos*	KeyboardYPos*
	Left	Name	Parent
	Prompt*	ShadowColor*	Tag
	TextColor*	Top	Wakeup*
	Events	Notify	

List Box

The Modular Windows list-box control is similar to the standard list-box control in Visual Basic. The following illustration shows the Toolbar icon and control image for the Modular Windows list box:



Icon



Control

Filename

TVLIST.VBX

Object Type

MWList

Remarks

If you change from a television font (TV large or TV small) to a Visual Basic font and reduce the size of the list box, you should issue a `form.cls` command for the list box.

Properties, Events, and Methods

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

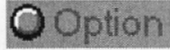
Group	Items		
Properties	About	Caption	DragIcon
	DragMode	Enabled	FaceColor*
	Font*	FontBold	FontItalic
	FontName	FontSelDelta	FontSize
	FontStrikethru	FontUnderline	FrameColor*
	Height	HightColor*	hWnd
	Index	Left	List
	ListCount	ListIndex	MousePointer
	Name	Parent	Reset*
	ShadowColor*	SelectString*	Sorted*
	TabIndex	TabStop	Tag
	Text*	TextColor*	Top
	TVdown*	TVleft*	TVright*
	TVup*	Visible	Width
	Events	Click	DragDrop
GotFocus		KeyDown	KeyPress
KeyUp		LostFocus	
Methods	AddItem	Drag	Move
	Refresh	RemoveItem	SetFocus

Option Button

The option button displays an option that can be turned on or off. The following illustration shows the Toolbar icon and control image for the Modular Windows option button:



Icon



Control

Filename

TVBTN.VBX

Object Type

MWOption

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Category	Items			
Properties	About	AlignOffset*	AlignStyle*	
	BackColor*	BitmapDown*	BitmapFace*	
	BitmapUp*	Caption	DragIcon	
	DragMode	Enabled	FaceColor*	
	Font*	FontBold	FontItalic	
	FontName	FontSize	FontStrikethru	
	FontUnderline	FrameColor*	Height	
	hWnd	HighltColor*	Index	
	Left	MousePointer	Name	
	Parent	Separation*	ShadowColor*	
	TabIndex	TabStop	Tag	
	TextColor*	Top	TransColorDown*	
	TransColorFace*	TransColorUp*	TransModeDown*	
	TransModeFace*	TransModeUp*	TVdown*	
	TVleft*	TVright*	TVup*	
	Value	Visible	Width	
	Events	Click	DragDrop	DragOver
		GotFocus	KeyDown	KeyPress
		KeyUp	LostFocus	
Methods	Drag	Move	Refresh	
	SetFocus			

Push Button

The Modular Windows push button performs a task when the user either clicks the button or presses the button's shortcut key. The following illustration shows the Toolbar icon and control image for the Modular Windows push button:



Icon

Control

Filename

TVBTN.VBX

Object Type

MWButton

Properties, Events, and Methods

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

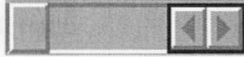
Category	Items			
Properties	About	AlignOffset*	AlignStyle*	
	BackColor*	BitmapDown*	BitmapFace*	
	BitmapUp*	Caption	DragIcon	
	DragMode	Enabled	FaceColor*	
	Font*	FontBold	FontItalic	
	FontName	FontSize	FontStrikethru	
	FontUnderline	FrameColor*	Height	
	hWnd	HighlightColor*	Index	
	Left	MousePointer	Name	
	Parent	Separation*	ShadowColor*	
	TabIndex	TabStop	Tag	
	TextColor*	Top	TransColorDown*	
	TransColorFace*	TransColorUp*	TransModeDown*	
	TransModeFace*	TransModeUp*	TVdown*	
	TVleft*	TVright*	TVup*	
	Value	Visible	Width	
	Events	Click	DragDrop	DragOver
		GotFocus	KeyDown	KeyPress
KeyUp		LostFocus		
Methods	Drag	Move	Refresh	
	SetFocus			

Scroll Bar (Horizontal)

The Modular Windows horizontal scroll bar responds to the hand-control device. The following illustration shows the Toolbar icon and control image for the Modular Windows horizontal scroll bar:



Icon



Control

Filename

TVSCROLL.VBX

Object Type

MWHScroll

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Group	Items		
Properties	About	BackColor*	BitmapThumb*
	ChannelFace*	ChannelFrame*	ChannelHight*
	ChannelShadow*	ChannelTextCol*	DisplayOnly*
	DragIcon	DragMode	Enabled
	Height	hWnd	Index
	Left	Max	Min
	MousePointer	Name	Parent
	Slider*	SmallChange*	TabIndex
	TabStop	Tag	ThumbFace*
	ThumbFrame*	ThumbHight*	ThumbInside*
	ThumbShadow*	ThumbTextCol*	Top
	TransModeColor*	TransModeThumb*	TVdown*
	TVleft*	TVright*	TVup*
	Value	Visible	Width
	Events	Change	DragDrop
GotFocus		LostFocus	Scroll
Methods	Drag	Move	Refresh
	SetFocus		

Scroll Bar (Vertical)

The Modular Windows vertical scroll bar responds to the hand-control device. The following illustration shows the Toolbar icon and control image for the Modular Windows vertical scroll bar:



Icon



Control

Filename

TVSCROLL.VBX

Object Type

MWVScroll

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Group	Items		
Properties	About	BackColor*	BitmapThumb*
	ChannelFace*	ChannelFrame*	ChannelHight*
	ChannelShadow*	ChannelTextCol*	DisplayOnly*
	DragIcon	DragMode	Enabled
	Height	hWnd	Index
	Left	Max	Min
	MousePointer	Name	Parent
	Slider*	SmallChange*	TabIndex
	TabStop	Tag	ThumbFace*
	ThumbFrame*	ThumbHight*	ThumbInside*
	ThumbShadow*	ThumbTextCol*	Top
	TransModeColor*	TransModeThumb*	TVdown*
	TVleft*	TVright*	TVup*
	Value	Visible	Width
	Events	Change	DragDrop
GotFocus		LostFocus	Scroll
Methods	Drag	Move	Refresh
	SetFocus		

Scroll Pad

The Modular Windows scroll pad responds to the Modular Windows hand control and provides a way to navigate around the display area. The following illustration shows the Toolbar icon and control image for the Modular Windows scroll pad:



Icon



Control

Filename TVPAD.VBX

Object Type MWPAD

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

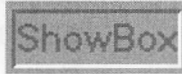
Group	Items		
Properties	About	BackColor*	DragIcon
	DragMode	Enabled	FaceColor*
	FrameColor*	Height	HighltColor*
	hWnd	Index	Left
	MousePointer	Name	PadStyle*
	Parent	ShadowColor*	TabIndex
	TabStop	Tag	TextColor*
	Top	TVdown*	TVleft*
	TVright*	TVup*	Visible
	Width		
Events	ClickDown	ClickLeft	ClickRight
	ClickUp	DragDrop	DragOver
	GotFocus	KeyDown	KeyPress
	KeyUp	LostFocus	
Methods	Drag	Move	Refresh
	SetFocus		

Show Box

The following illustration shows the Toolbar icon and control image for the Modular Windows show box:



Icon



Control

Filename TVSTA.VBX

Object Type MWShowbox

Properties, Events, and Methods

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Group	Items			
Properties	About	AlignOffset*	AlignStyle*	
	BitmapFace*	Caption	DragIcon	
	DragMode	Enabled	FaceColor*	
	Font*	FontBold	FontItalic	
	FontName	FontSize	FontStrikethru	
	FontUnderline	FrameColor*	Height	
	HighltColor*	hWnd	Index	
	Left	MousePointer	Name	
	NoEdge*	NoFrame*	Parent	
	RectStyle*	Separation*	ShadowColor*	
	TabIndex	Tag	TextColor*	
	TextHAlign*	TextVAlign*	Top	
	TransColorFace*	TransModeFace*	Visible	
	Width			
	Events	DragDrop	DragOver	MouseDown
		MouseMove	MouseUp	
Methods	Refresh	Move	Drag	

Sticky Button

The sticky button control is similar to the push button. However, the sticky button changes state only when the user presses the hand-control action button. Unlike the push button, the sticky button does not automatically return to the up state. The user must press the hand-control action button to change the button's state. The following illustration shows the Toolbar icon and control image for the Modular Windows sticky button:



Icon



Control

Filename TVBTN.VBX

Object Type MWSticky

**Properties,
Events, and
Methods**

All of the properties, events, and methods for this control are listed below. Properties marked with an asterisk (*) are further documented later in this chapter.

Category	Items			
Properties	About	AlignOffset*	AlignStyle*	
	BackColor*	BitmapDown*	BitmapFace*	
	BitmapUp*	Caption	DragIcon	
	DragMode	Enabled	FaceColor*	
	Font*	FontBold	FontItalic	
	FontName	FontSize	FontStrikethru	
	FontUnderline	FrameColor*	Height	
	hWnd	HighltColor*	Index	
	Left	MousePointer	Name	
	Parent	Separation*	ShadowColor*	
	TabIndex	TabStop	Tag	
	TextColor*	Top	TransColorDown*	
	TransColorFace*	TransColorUp*	TransModeDown*	
	TransModeFace*	TransModeUp*	TVdown*	
	TVleft*	TVright*	TVup*	
	Value	Visible	Width	
	Events	Click	DragDrop	DragOver
		GotFocus	KeyDown	KeyPress
		KeyUp	LostFocus	
	Methods	Drag	Move	Refresh
SetFocus				

Properties Reference

This is an alphabetic reference to the non-standard properties of the Modular Windows Visual Basic Custom Controls. Use this material as you would that which is documented in the *Microsoft Visual Basic Custom Control Reference*.

AlignOffset

Applies To	Check box, group box, option button, push button, show box, sticky button
Description	This property specifies the amount of offset between the edge of the control and the bitmap used on the control. See AlignStyle for related information.
Usage	<code>[form.]objecttype.AlignOffset[=setting%]</code>
Settings	You must use integer values as settings for this property. The value assigned to this property specifies a number of pixels. The maximum values depend on the resolution of the monitor or display device. To determine the maximum value, subtract the control's horizontal and vertical dimensions from the width and height of the television's display area.
Data Type	Integer

AlignStyle

Applies To	Check box, group box, option button, push button, show box, sticky box
Description	This property specifies the alignment of the bitmap on a control.
Usage	<code>[form.]objecttype.AlignStyle[=setting%]</code>
Settings	The settings for this property are as follows:

Setting	Description
0	Bitmap left: bitmap is aligned with the left edge of the control.
1	Bitmap center: bitmap is centered on the control.
2	Bitmap right: bitmap is aligned with the right edge of the control.

Continued

Setting	Description
3	Bitmap top (show box): bitmap is aligned with the top of the control. Align left (all other controls): text and bitmaps on the control are aligned at the left edge of the control.
4	Bitmap bottom (show box): bitmap is aligned with the bottom of the control. Align right (all other controls): text and bitmaps on the control are aligned at the right edge of the control.

Data Type Integer (Enumerated)

BackColor

Applies To Check box, group box, list box, option button, push button, sticky button, horizontal scroll bar, vertical scroll bar, scroll pad

Description This property defines the background color for the controls specified. By setting this property to the same color you use in the display area, you can hide the square corners by the rounded edges on the list box, push button, and sticky button.

Usage *[form.]objecttype.BackColor[=*setting*&]*

Settings This property uses the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of Settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QBColor functions in code.
System default colors	Colors specified with system color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

BitmapDown, BitmapFace, and BitmapUp

Applies To Check box, group box (BitmapFace only), option button, push button, show box (BitmapFace only), sticky button

Description These properties control the appearance of the button. They support bitmaps only, not icons or metafiles.

- BitmapDown is the bitmap for the button's down state.
- BitmapFace is the bitmap for the button text area. For static labels, edit boxes and show boxes, BitmapFace is the bitmap for the static text area.
- BitmapUp is the bitmap for the button's up state.

Usage *[form.]objecttype.BitmapDown=LoadPicture([stringexpression\$])*

[form.]objecttype.BitmapFace=LoadPicture([stringexpression\$])

[form.]objecttype.BitmapUp=LoadPicture([stringexpression\$])

Settings The following settings are valid for these properties:

Setting	Description
(none)	No picture (default).
(Bitmap)	Designates a graphic to display. You can load the graphic from the Properties list box at design time. At run time, you can also set the property by using the LoadPicture function on a bitmap.

At design time, you can use the Load Picture dialog box to select a graphic. To open the Load Picture dialog box, double-click the property name in the Properties list box. At run time, you can use Clipboard methods such as **GetData**, **SetData**, and **GetFormat** with the non-text Clipboard formats CF_BITMAP, and CF_DIB, as defined in CONSTANT.TXT.

When setting these properties at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic is not saved with the application, so you must save the picture file for the user along with the executable file.

Note At run time these properties can be set to any other object's DragIcon, Icon, Picture, or Image property, or you can assign it the graphic returned by the **LoadPicture** function. These properties can only be assigned directly.

Use the **SavePicture** statement to save a graphic from a form or picture box into a file.

Data Type Picture

BitmapThumb

Applies To Horizontal scroll bar, Vertical scroll bar

Description Allows you to select a bitmap to use as the thumb.

Usage *[form.]objecttype.BitmapThumb=LoadPicture([stringexpression\$])*

Settings The following settings are valid for these properties:

Setting	Description
(none)	No picture (default).
(Bitmap)	Designates a graphic to display. You can load the graphic from the Properties list box at design time. At run time, you can also set the property by using the LoadPicture function on a bitmap.

At design time, you can use the Load Picture dialog box to select a graphic. To open the Load Picture dialog box, double-click the property name in the Properties list box. At run time, you can use Clipboard methods such as **GetData**, **SetData**, and **GetFormat** with the non-text Clipboard formats CF_BITMAP, and CF_DIB, as defined in CONSTANT.TXT.

When setting these properties at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic is not saved with the application, so you must save the picture file for the user along with the executable file.

Note At run time these properties can be set to any other object's DragIcon, Icon, Picture, or Image property, or you can assign it the graphic returned by the **LoadPicture** function. These properties can only be assigned directly.

Use the **SavePicture** statement to save a graphic from a form or picture box into a file.

Data Type Picture

ChannelFace, ChannelFrame, ChannelHight, ChannelShadow, ChannelTextCol

Applies To Horizontal scroll bar, vertical scroll bar

Description These properties assign colors to the face, frame, highlight, shadow, and text column of the scroll bar.

Usage *[form.]objecttype.ChannelFace[=setting&]*
 [form.]objecttype.ChannelFrame[=setting&]
 [form.]objecttype.ChannelHight[=setting&]
 [form.]objecttype.ChannelShadow[=setting&]
 [form.]objecttype.ChannelTextCol[=setting&]

Settings These properties use the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QColor functions in code.
System default colors	Colors specified with system-color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

DefaultKey

Applies To Edit box, keyboard

Description This property specifies which key initially gets the focus when the Keyboard is activated. The case of the specified letter determines whether the caps lock function is on or off when the keyboard is activated.

For example, if the designated letter is "a", the "a" key will get the focus when the keyboard is activated, and the caps-lock function will be off. If the designated letter is "A", the "a" key will still get the focus, but the caps-lock function will be on.

Usage *[form.]objecttype.DefaultKey[=setting\$]*

Settings Any string is a valid setting for this property. The first character specifies the focus key. The case of the first character determines the state of the caps-lock function.

Data Type String

DefaultLength

Applies To Edit box, keyboard

Description This property specifies the maximum keyboard buffer length.

Usage *[form.]objecttype.DefaultLength[=setting%]*

Settings	The maximum allowable setting for this property depends on the amount of memory available in the system. To specify a value, enter the full value you want to use. For example, to use a 1K buffer, enter 1024.
Data Type	Integer

DefaultText

Applies To	Edit box, keyboard
Description	This property defines the default text to assign to a control. If no value is assigned to this property, the value of Name will be used at design time, and nothing will be used at run time.
Usage	<code>[form.]objecttype.DefaultText[=setting\$]</code>
Settings	You can assign any string as the value for this property.
Data Type	String

DisplayOnly

Applies To	Horizontal scroll bar, vertical scroll bar
Description	Disables or enables the spin buttons at the ends of the scroll bar.
Usage	<code>[form.]objecttype.DisplayOnly[=setting%]</code>
Settings	True (-1) or False (0)
Data Type	Integer (Boolean)

FaceColor, FrameColor, HightColor, ShadowColor, and TextColor

- Applies To** Check box, edit box, group box, popup keyboard, list box, option button, push button, scroll pad, show box, sticky button
- Description** These properties determine the color of a button's face, highlight, shadow, frame, and text, as follows:
- FaceColor is an internal property for the static label.
 - For the edit box, the FrameColor property affects only the frame of the popup keyboard used with the edit box. FrameColor does not affect the frame of the edit box.

Usage

[form.]objecttype.FaceColor[=setting\$]

[form.]objecttype.HightColor[=setting\$]

[form.]objecttype.ShadowColor[=setting\$]

[form.]objecttype.FrameColor[=setting\$]

[form.]objecttype.TextColor[=setting\$]

- Settings** These properties use the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.
- The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QBColor functions in code.
System default colors	Colors specified with system-color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

Font

Applies To	Check box, edit box, group box, list box, option button, push button, show box, static label, sticky button								
Description	<p>The Font property determines the font used to display text on the television screen.</p> <p>This property determines whether the six standard Visual Basic font properties (FontBold, FontItalic, FontName, FontSize, FontStrikethru, and FontUnderline) are active.</p> <p>The standard properties are active when the Font property has a value of 2. If the Font property is set to 0 or 1, any attempts to set the standard font properties will fail without error, and the related properties will display the information using the Visual Basic font that most closely matches the currently selected TV font.</p>								
Usage	<i>[form.]objecttype.Font[=setting%]</i>								
Settings	The settings for this property are as follows:								
	<table border="1"> <thead> <tr> <th>Setting</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Selects the default small TV font</td> </tr> <tr> <td>1</td> <td>Selects the default large TV font</td> </tr> <tr> <td>2</td> <td>Permits the other font properties to select the font</td> </tr> </tbody> </table>	Setting	Description	0	Selects the default small TV font	1	Selects the default large TV font	2	Permits the other font properties to select the font
Setting	Description								
0	Selects the default small TV font								
1	Selects the default large TV font								
2	Permits the other font properties to select the font								
Data Type	Integer (Enumerated)								

FontSelDelta

Applies To	List box
Description	This property determines the increase in font height for the entry that is in the selection window of the list box. This property has no effect unless the Font property is set to 2 (to use Visual Basic fonts).

Note If the Font property is set to 0 (to use the TV small font), Modular Windows will use the standard small TV font when the control is not selected, and will use the standard large TV font when the control is selected.

If the Font property is set to 1 (to use the TV large font), Modular Windows will use the standard large TV font when the control is selected as well as when it is not selected.

Usage	<i>[form.]objecttype.FontSelDelta</i> [= <i>setting%</i>]
Settings	Any integer is a valid setting for this property.
Data Type	Integer

KbdHwnd

Applies To	Keyboard
Description	This property is the Windows handle of the popup keyboard window. It is not applicable to the control itself, which is hidden at run time. This is a read-only property.
Usage	<i>[form.]objecttype.KbdHwnd</i> [= <i>setting%</i>]
Settings	Not applicable; set by Modular Windows.
Data Type	Integer

KeyboardXPos

Applies To	Edit box, keyboard
Description	The X (horizontal) screen-coordinate of the upper-left corner of the control.
Usage	<i>[form.]objecttype.KeyboardXPos</i> [= <i>setting%</i>]
Settings	The value assigned to this property is interpreted as a number of pixels on the screen. The maximum value is the same as the width of the television's display area.
Data Type	Integer

KeyboardYPos

Applies To	Edit box, keyboard
Description	The Y (vertical) screen-coordinate of the upper-left corner of the control.
Usage	<i>[form.]objecttype.KeyboardYPos[=setting%]</i>
Settings	The value assigned to this property is interpreted as a number of pixels on the screen. The maximum value is the same as the vertical measurement in pixels of the television's display area.
Data Type	Integer

Name

Applies To	All custom controls
Description	Sets the identifier used to access the control in code; not available at run time. By default, the name for this property includes the control name and an integer, yielding a unique name such as "tvOption1."
Usage	<i>[form.]objecttype.Name[=setting%]</i>
Settings	A Name must start with a letter and cannot be longer than 40 characters when combined with an event name, including alphanumeric or underscore characters. A Name cannot be a reserved word. Note that Name is distinct from other properties that label or display a control's contents at run time, such as the Caption, Text, or Value properties.
Data Type	String

NoEdge

Applies To	Show box
Description	This property controls the 3-D edge effect around the edge of the control.
Usage	<i>[form.]objecttype.NoEdge[=setting%]</i>

Settings The settings for this property are True and False, as follows:

Setting	Description
True	Edge is hidden
False	Edge is visible

Data Type Integer (Boolean)

NoFrame

Applies To Show box

Description This property determines whether a control's frame is hidden or visible.

Usage `[form.]objecttype.NoFrame[=setting%]`

Settings The settings for this property are True and False, as follows:

Setting	Description
True	Frame is hidden
False	Frame is visible

Data Type Integer (Boolean)

PadStyle

Applies To Scroll pad

Description This property determines the type of scroll pad displayed.

Usage `[form.]objecttype.PadStyle[=setting%]`

Settings The settings for the PadStyle property are as follows:

Setting	Description
0	Full Scroll pad
1	Horizontal scroll pad only
2	Vertical scroll pad only

Data Type Integer (Enumerated)

Prompt

Applies To Keyboard

Description This property sets the prompt caption of the keyboard.

Usage *[form.]objecttype.Prompt[=setting\$]*

Settings Any string value.

Data Type String

RectStyle

Applies To Show box

Description This property determines the style of rectangle used for the control.

Usage *[form.]objecttype.RectStyle[=setting%]*

Settings The settings for the RectStyle property are as follows:

Setting	Description
0	Up rectangle
1	Down rectangle

Data Type Integer (Enumerated)

Reset

Applies To	List box
Description	Clears the contents of a list box. This property can only be used at run time.
Usage	<i>[form.]objecttype.Reset[=setting%]</i>
Settings	You can assign any integer to this property. Assigning any non-null value to the property resets the control. For example, <code>MWList.Reset = 0</code> resets the list box.
Data Type	Integer

SelectString

Applies To	List box
Description	This run-time, write-only property implements the <code>LB_SELECTSTRING</code> message. The array index specifies the beginning search location, and a search is made for the string specified in the message. If the string is found, it will be selected.
Usage	<i>[form.]objecttype.SelectString[=setting\$]</i>
Settings	Any string value.
Data Type	String

Separation

Applies To	Check box, Group box, Option button, Push button, Show box, Sticky button
Description	This property specifies the text spacing on the face bitmap of a control.
Usage	<i>[form.]objecttype.Separation[=setting%]</i>
Settings	The units for this property are screen pixels.
Data Type	Integer

Slider

Applies To	Horizontal scroll bar, Vertical scroll bar
Description	This property determines whether the scroll bar is configured as a slider.
Usage	<code>[form.]objecttype.Slider[=setting%]</code>
Settings	0 = False, 1 = True
Data Type	Integer (Boolean)

SmallChange

Applies To	Horizontal scroll bar, Vertical scroll bar
Description	Specifies the distance the thumb moves after the user clicks an arrow in a scroll bar. Setting this property effectively sets the Value property of the thumb.
Usage	<code>[form.]objecttype.SmallChange[=setting%]</code>
Settings	Valid settings must be within the range of the current settings of Max and Min.
Data Type	Integer

Sorted

Applies To	List box
Description	Determines whether the elements of a list box are automatically sorted alphabetically. Setting this property to TRUE clears the list box. You must later reinitialize the list box to use it.

Note In Visual Basic Extensions, this property is read-write at run time. This is not true in Visual Basic 2.0, where this property is read-only at run time.

For more information on this property, see the *Microsoft Visual Basic Custom Control Reference*.

Usage	<i>[form.]objecttype.Sorted</i> [= <i>setting%</i>]						
Settings	The following settings are valid for the Sorted property:						
	<hr/>						
	<table><thead><tr><th>Setting</th><th>Description</th></tr></thead><tbody><tr><td>True (-1)</td><td>Keeps list items sorted alphabetically</td></tr><tr><td>False (0)</td><td>Specifies no sorting of list items (default)</td></tr></tbody></table> <hr/>	Setting	Description	True (-1)	Keeps list items sorted alphabetically	False (0)	Specifies no sorting of list items (default)
Setting	Description						
True (-1)	Keeps list items sorted alphabetically						
False (0)	Specifies no sorting of list items (default)						
Data Type	Integer (Boolean)						

Text

Applies To	Edit box, List box
Description	<p>This property can only be used at run time. It gets and sets the contents of the control. At run time, Visual Basic copies the value from DefaultText property into the Text property.</p> <p>The Text property is primarily a read property, used to capture the text entered by the user.</p>
Usage	<i>[form.]objecttype.Text</i> [= <i>setting%</i>]
Settings	Any string can be used as a value for this property. The maximum length depends on the size of the control.
Data Type	String

TextHAlign

Applies To	Show box
Description	Determines the horizontal alignment of text in the control.
Usage	<i>[form.]objecttype.TextHAlign</i> [= <i>setting%</i>]

Settings The settings for this property areas follows:

Setting	Description
0	Aligns text near left edge of control
1	Aligns text near right edge of control
2	Aligns text near center of control

Data Type Integer

TextVAlign

Applies To Show box

Description Determines the vertical alignment of text in the control.

Usage *[form.]objecttype.TextVAlign[=setting%]*

Settings The settings for this property are the following:

Setting	Description
0	Top
1	Center
2	Bottom

Data Type Integer

ThumbFace, ThumbFrame, ThumbHight, ThumbShadow, ThumbTextCol

Applies To Horizontal scroll bar, vertical scroll bar

Description These properties assign colors to the face, frame, highlight, shadow, and text column of the scroll bar.

Usage

`[form.]objecttype.ThumbFace[=setting&]`
`[form.]objecttype.ThumbFrame[=setting&]`
`[form.]objecttype.ThumbHight[=setting&]`
`[form.]objecttype.ThumbShadow[=setting&]`
`[form.]objecttype.ThumbTextCol[=setting&]`

Settings

These properties use the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QBColor functions in code.
System default colors	Colors specified with system-color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

ThumbInside

Applies To Horizontal scroll bar, vertical scroll bar

Description Determines whether the thumb is inside the border of the scroll bar.

Usage `[form.]objecttype.ThumbInside[=setting%]`

Settings True (-1) or false (0).

Data Type Integer (Boolean)

TransColorDown, TransColorFace, TransColorUp

Applies To Check box, group box (TransColorFace only), option button, push button, show box (TransColorFace only), sticky button

Description Specifies that a certain color should be transparent in a bitmap used on a control. For example, if yellow is designated as TransColorFace and portions of the bitmap are yellow, those portions will not be painted when the bitmap is drawn.

These colors are used only when TransModeDown, TransModeFace, and TransModeUp are set to 1. When TransModeDown, TransModeFace and TransModeUp are set to zero, these colors have no effect.

Usage *[form.]objecttype.TransColorDown[=setting&]*

[form.]objecttype.TransColorFace[=setting&]

[form.]objecttype.TransColorUp[=setting&]

Settings These properties use the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QBColor functions in code.
System default colors	Colors specified with system-color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

TransColorThumb

Applies To Horizontal scroll bar, vertical scroll bar

Description This property defines the transparent color for the thumb bitmap.

This color is used only when TransColorMode is set to 1. If TransColorMode is set to zero, TransColorThumb has no effect.

Usage *[form.]objecttype.TransColorThumb[=setting&]*

Settings This property uses the Microsoft Windows RGB scheme for colors. To display the palette of available colors, double-click the property name in the Properties list box.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0. The lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Each property accepts the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, the RGB scheme, or QBColor functions in code.
System default colors	Colors specified with system-color constants from CONSTANT.TXT, a Visual Basic file you can load into a project's global module. Windows substitutes the user's choices, as specified through the user's Control Panel settings.

Data Type Long

TransModeDown, TransModeFace, and TransModeUp

Applies To Check box, group box (TransModeFace only), option button, push button, show box (TransModeFace only), sticky button

Description These properties control the background mode (transparent or opaque) for the bitmaps associated with a button, as follows:

- TransModeDown is the background mode for BitmapDown.
- TransModeFace is the background mode for BitmapFace.
- TransModeUp is the background mode for BitmapUp.

Usage `[form.]MWCheckBox.TransModeDown[=setting%]`

`[form.]MWCheckBox.TransModeFace[=setting%]`

`[form.]MWCheckBox.TransModeUp[=setting%]`

Settings The settings for these properties are as follows:

Setting	Description
0	Opaque (default)
1	Transparent

When these properties are set to 1, the transparent color is determined by TransColorMode.

Data Type Integer (Enumerated)

TransModeThumb

Applies To Horizontal scroll bar, vertical scroll bar

Description This property controls the mode (transparent or opaque) of the thumb in the scroll bar.

Usage `[form.]objecttype.TransModeThumb[=setting%]`

Settings The settings for these properties are as follows:

Setting	Description
0	Opaque (default)
1	Transparent

When this property is set to 1, the transparent colors are determined by TransColorDown, TransColorFace, and TransColorDown.

Data Type Integer (Enumerated)

TVdown, TVleft, TVright, TVup

Applies To Check box, edit box, group box, list box, option button, push button, scroll bar (vertical), scroll bar (horizontal), scroll pad, sticky button

Description These properties set the direction for the focus manager direction array. They are run-time, write-only properties, and accept a window handle as a parameter.

- TVdown determines where the focus goes when the Down direction button is pressed.
- TVleft determines where the focus goes when the Left direction button is pressed.
- TVright determines where the focus goes when the Right direction button is pressed.
- TVup determines where the focus goes when the Up direction button is pressed.

Ordinarily, you should let the Focus Manager control the responses to direction keys. If Focus Manager does not produce the action you expect based on the Modular Windows style guidelines, you can use these properties to adjust it. If you use these properties, make sure the action you define is in agreement with the style guidelines.

For more information about the style guidelines, see the Design Guide in the Modular Windows SDK.

Usage	<code>[form.]objecttype.TVdown[=setting%]</code> <code>[form.]objecttype.TVleft[=setting%]</code> <code>[form.]objecttype.TVright[=setting%]</code> <code>[form.]objecttype.TVup[=setting%]</code>
Settings	<p>If you assign zero (0) to these properties, the focus manager will control the movement of the focus automatically.</p> <p>To select a particular control, set the property value to the hWnd value of the desired control.</p>
Data Type	Integer

Wakeup

Applies To	Keyboard
Description	This is a write-only, run-time property. Assigning any value to this property causes the keyboard to appear.
Usage	<code>[form.]objecttype.Wakeup[=setting%]</code>
Settings	This property accepts any integer value.
Data Type	Integer

Online TV Program Guide

As digital and fiber-optic technology bring more viewing channels to our televisions, convenient methods to organize and access the many program and movie options will become valuable services. This Online TV Program Guide simulates an online TV listing that helps TV viewers preview and select programs and movies they want to watch.

Note The Online TV Program Guide does not run on the Tandy VIS system. This program is a conceptual model of the type of programs that will soon be available for Modular Windows platforms other than the Tandy VIS.

This application has been developed with Microsoft Visual Basic 2.0. Although this prototype runs under Microsoft Windows 3.1, it uses the Modular Windows TVUI controls and was written to suggest one type of application that can be developed for future Modular Windows platforms. Modular Windows running on cable boxes, satellite receivers, TVs, and VCRs opens up the possibility of many unique applications and demonstrates the reality of enhanced television using existing Microsoft tools.

The Modular Windows SDK includes this application to help companies developing Microsoft Modular Windows applications envision and experiment with interactive television and online applications for subsequent releases of Modular Windows. The SDK provides all Visual Basic 2.0 source files and controls used by the program. The “live” TV footage is simulated using a Microsoft Video for Windows CD-ROM file. An actual application would process live-cable video input. The following section describes the Video for Windows technology used in this prototype.

Program Features

The Online TV Program Guide has three screens. The Movie Previews screen is the first screen you see when running the application. The Program Guide screen lists all TV shows at a particular time, and the Program Topics screen sorts available shows by topic.

From the Movie Previews screen, you can watch previews of upcoming movies and order pay-per-view movies using a simple impulse button on the online order form.

Using a pointing device, you can browse through a list of upcoming shows organized by time or channel on the Online TV Program Guide screen. The Online TV Program Guide interface displays the selected show in a small TV viewing box.

You can switch from the small TV window to watch the show on full-screen TV display by choosing the TV button at the bottom of the Program Guide and Program Topics screens. (This feature of the demo is implemented only for systems with DVI video hardware installed. For information about DVI video hardware, see the following section, “Hardware and Software Requirements.”)

The Program Topics screen shows available programs sorted by topics such as sports, news, movies, or comedies. The buttons used to access other parts of the program are displayed at the bottom of the screen after you select one of the topics.

A Microsoft Video For Windows 1.0 AVI file simulates actual TV footage. Video for Windows allows you to play synchronized full-motion digital-video sequences on a PC without specialized hardware. Video for Windows interleaves video frames with audio data within the file containing the video sequence. The term *interleaved* refers to the way video and audio are alternatively stored in a video file. Video for Windows stores files in a format that supports interleaving and is called the Audio/Video Interleaved (AVI) format. Video files are similar to traditional movies. They contain frames of image data that are displayed sequentially and played concurrently with a sound track. In a video file, audio and video data are stored together.

Without an Intel® Indeo™ video card for DVI video support or other third-party DVI hardware, Video for Windows displays video sequences at 15 frames per second at a 160-by-120 resolution in the small TV display window. Performance deteriorates in the larger movie previews 320-by-240 display. With DVI hardware, however, all resolutions and display sizes execute at 15 frames per second. See the following section, “Hardware and Software Requirements,” for additional information about DVI video hardware.

Important Because the Online TV Program Guide includes actual television footage, strict conditions must be met for use of this demo. The television footage is cleared only for playback with this demo.

For more information on licensing and clearance, see the last section of this chapter and your licensing agreement.

The application could be easily expanded to provide additional features. See “Program Modification,” later in this chapter, for more information.

Hardware and Software Requirements

The following list gives the hardware requirements for running the Online TV Program Guide. A Tandy VIS system is not required.

- An 80386 CPU or later.
- 4 MB of memory.
- 1 MB of available disk space to install demo software. To run AVI video files from CD-ROM, no additional disk space is needed. Installing video files on a hard disk improves AVI performance but requires an additional 40 MB of disk space.
- A CD-ROM drive.
- A SVGA video driver that supports 640-by-480 resolution and 256 colors.
- A mouse.
- Intel Indeo card for DVI video (optional).

The Intel Indeo card improves the performance of the demo and allows full-screen TV viewing from the Program Guide or Program Topics screens.

In addition to the hardware listed above, you need certain software installed on your system before running the demo. Software requirements include the following:

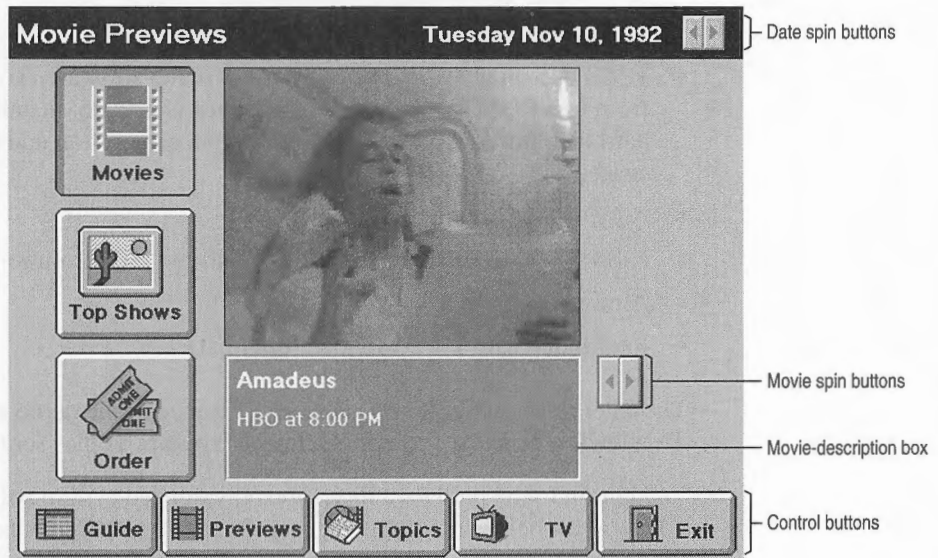
- MS-DOS version 5.0 or later.
- Microsoft Windows 3.1.
- Visual Basic version 2.0.
- Video for Windows 1.0 run time.

Installation and Operation

The Modular Windows SDK makes it easy for you to load and run the Online TV Program Guide. The Setup program asks if you have DVI video hardware installed. If you have DVI video hardware such as the Intel Indeo card, the demo will make use of it, allowing faster video playback than the software-only mode of AVI playback.

Setup then installs all the necessary files on your computer and adds an icon to the Visual Basic for Modular Windows program group that you can use to start the program.

Double-clicking the icon in the Visual Basic for Modular Windows program group loads the demo. When you run the Online TV Program Guide application, the Movie Previews screen appears. The following illustration shows the Movie Previews screen for the Online TV Program Guide:



Amadeus, ©1983, The Saul Zaentz Company.
All rights reserved.

The viewing screen cycles through previews of upcoming hit movies. The show title and the time this show begins appears beneath the window.

The control buttons at the bottom of the screen provide access to other screens in the application. The Guide button and the Topics button take you to the Program Guide screen and Program Topics screen explained below. The TV button takes you to full-screen TV display.

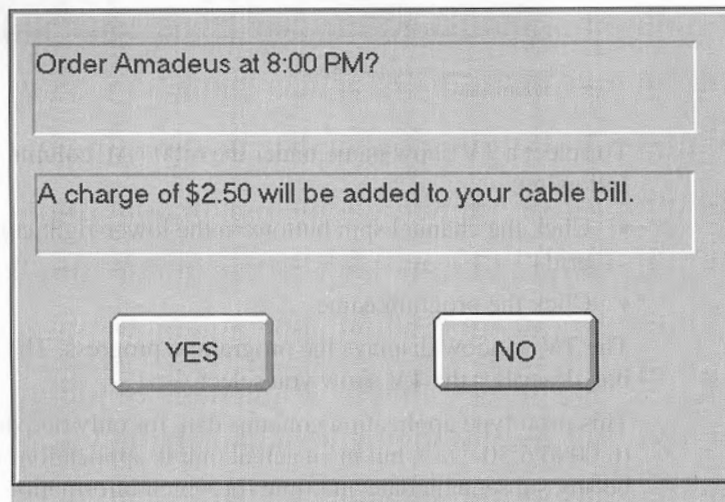
► **To explore the features of the Online TV Program Guide:**

1. Double-click the icon the Visual Basic for Modular Windows program group.
2. To view available movie previews, click on the spin buttons to the right of the description box.
3. To display a list of upcoming popular shows, choose the Top Shows button.

This demo does not provide a list of top shows, but this feature can be implemented so you can preview shows selected by the cable station.

4. To order a pay-per-view movie, choose the Order button to display an order form.

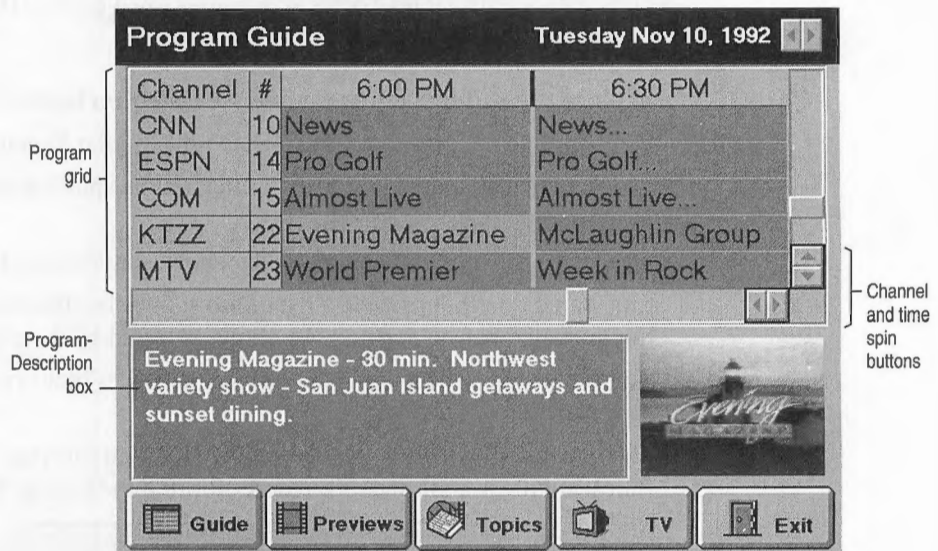
The cost for viewing the movie is charged directly to your cable bill. The following illustration shows the order form in the Online TV Program Guide:



The screenshot shows a dialog box with a light gray background. At the top, a text box contains the question "Order Amadeus at 8:00 PM?". Below this, another text box displays "A charge of \$2.50 will be added to your cable bill." At the bottom of the dialog, there are two buttons: "YES" on the left and "NO" on the right. Both buttons have a 3D effect with a shadow.

5. Choose either the Yes or No button to close the order form.

6. Next, choose the Guide button in the lower-left corner of the Movie Previews screen. The following Program Guide screen appears:



Evening Magazine, ©1992, King5 Television.
All rights reserved.

7. To select a TV show name under the 6:00 P.M. column, do one of the following:
- Click the channel spin buttons in the lower-right corner of the program grid.
 - Click the program name.

The TV window displays the program in progress. The program description box describes the TV show you select.

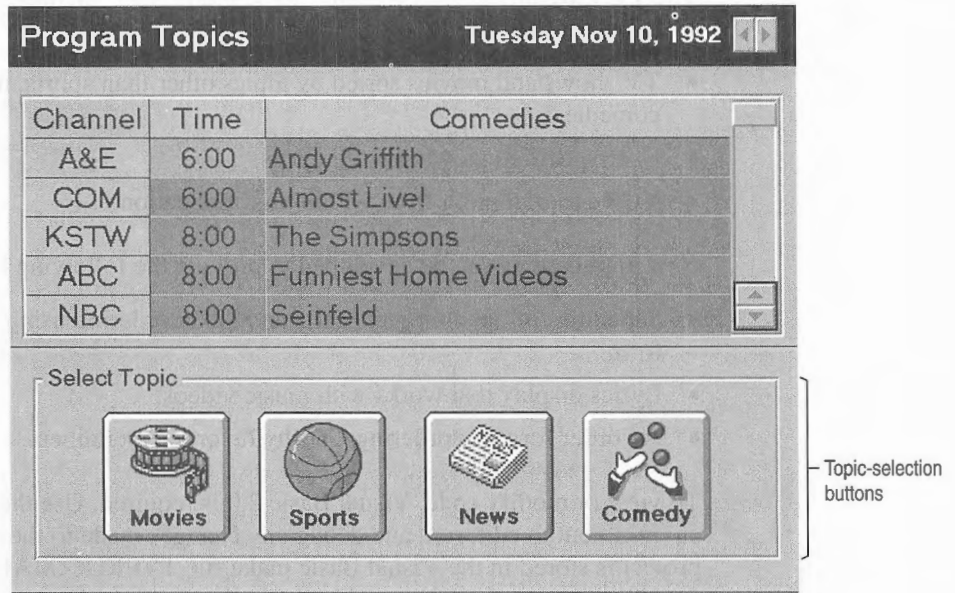
This prototype application contains data for only one day and one time slot (6:00 to 6:30 P.M.), but in an actual online application, the date and time could be any subsequent date and time for which information is available.

8. To switch to full-screen viewing, choose the TV button in the lower-right corner of the screen.

Full-screen display is available only on systems with DVI video hardware.

9. To return from full-screen TV viewing to the program grid, click in the full-screen TV display window.

10. To move to the Program Topics screen, choose the Topics button.
The following Program Topics screen appears:



Selecting a topic displays a list of TV shows within each category.

11. To move through the list of TV shows, do one of the following:
 - Select the spin buttons on the right side of the program grid.
 - Click the program name.
12. To select a different topic, choose the Topics button again.
After selecting a topic, the topic buttons are replaced by the same buttons shown at the bottom of the Program Guide or Movie Previews screens.
13. Choose the Guide button or the Previews button to return to either of those screens.
14. To exit the program, choose the Exit button.

Program Modification

The TV Program Guide program can be modified to include any of the following features:

- TV shows and movies sorted by topics other than sports, movies, news, and comedies.
- User-specified viewing preferences.
- VCR programming from within this application.

New applications can be developed to provide the following kinds of services:

- Schedules of upcoming concerts and ticket ordering using an online order form.
- Lyrics display that works with music videos.
- An order form for ordering CDs by favorite entertainers.

To view or modify code, Visual Basic 2.0 is required. Use the Visual Basic 2.0 environment to edit, run, and debug any changes made to the cable code. The project is stored in the Visual Basic make file TVPROG.MAK. The following list describes the required files and how they are used by the program:

Filename	Description
BARKER.BAS	Contains procedures related to the Movie Previews screen.
BARKER.FRM	The initial startup Movie Previews screen.
BARKER.FRX	Contains binary code for BARKER.FRM that cannot be saved in text format.
BARKER.TXT	A data file that supplies information for the Movie Previews screen. The following information is kept for each selection: Number(#)—sequential Title—show title Show times—sentence describing show times Key—key name (must exist in VDACCESS.TXT)
CHOOSE.FRM	The screen displayed to allow the user to choose a category on the Program Topics screen.
CHOOSE.FRX	Contains binary code for CHOOSE.FRM that cannot be saved in text format.
ORDER.FRM	The screen accessed from the Movie Previews screen that allows the user to order the current movie selection.

Continued

Filename	Description
TVGRID.VBX	Visual Basic 2.0 grid control used to display the grid of program information.
GRIDSUP.BAS	Contains procedures that simplify interfacing with a grid control.
MCI.VBX	Visual Basic 2.0 multimedia control used to access and manipulate AVI files.
MSG.FRM	Provides placeholder text “No previews of Top Shows are available.”
README.TXT	Contains any new information provided after documentation publication.
SHOWGRD.TXT	Contains the information displayed in the program grid. Cells of columns contain a time heading known as <i>time slots</i> . For each time slot, three hidden columns exist to the right of the time slot. The columns, in order from left to right, are as follows: the program description, the program length, and the name of a video selection to be played when the cell is highlighted. Fields in SHOWGRD.TXT are tab delimited. This file can be edited using a spreadsheet program, such as Microsoft Excel.
TOPICS.BAS	Contains procedures related to the Program Topics screen.
TOPICS.FRM	The screen that allows the user to selectively view categories of type: movies, sports, news, or comedies.
TOPICS.TXT	The data file supplying information to the topics screen. The following information is kept for each selection: Number(#)—sequential Time—specific show time Channel—ABC, NBC, etc. Title—show title Category—either movies, sports, news, or comedies Subcategory—not used Desc—a short description of the show Key—unique key name (must exist in VDACCESS.TXT)

Continued

Filename	Description
TVPROG.INI	A standard Microsoft Windows initialization file that specifies the path and name of the video file to be used by the application. This file must be located in the (Windows) default directory.
TVPROG.FRM	A Visual Basic 2.0 form that contains all the controls for the Online TV Program Guide application.
TVPROG.BAS	Contains application global variables and types as well as Visual Basic 2.0 and Windows constants. Application constants appear first, followed by Visual Basic constants, followed by Windows API declarations and constants. Also contains the “main” procedure for the application.
TVPROG.EXE	The TV Program Guide executable file.
TVPROG.MAK	The Visual Basic 2.0 project file, which is also a text file.
TVPROG.FRX	Contains binary information for either TVPROG.FRM or its controls that cannot be saved in text format.
TVMSV.AVI or TVDVI.AVI	Contains all the AVI clips used by the TV Program Guide and can be viewed using any application that supports AVI. Setup installs TVDVI.AVI if your system has DVI hardware. Otherwise, Setup installs the TVMSV.AVI file. Setup also edits the TVPROG.INI file to point to the correct file and its location.
UTIL.BAS	Contains miscellaneous support procedures used within the TV Program Guide.
VDAMSV.TXT VDADV1.TXT	<p>A table of information used to specify the location of each video clip in the AVI file. The following information is kept for each selection:</p> <p>Item Number—the number of the entry in the table; each entry must start with a number</p> <p>Description—a textual description of the program</p> <p>Key—a word (that cannot contain spaces) used to find the selection in the table at run time</p> <p>Start—the frame number of the beginning of the selection</p> <p>End—the frame number of the end of the selection</p> <p>At run time, procedures are called to display a specific video segment by name. These are tab-delimited files, which can be viewed or modified using a spreadsheet program, such as Microsoft Excel.</p>

Redistribution Rights

Because the Online TV Program Guide uses actual TV footage in the AVI file, the conditions for using the Online TV Program Guide are strict. You should be aware of the following rules of usage:

- The television footage can only be used for playback with the Online TV Program Guide demo.
- Any public use of this demo must be secured from copyright holders **before** using this demo.
- Any other use must be secured from copyright holders in advance.

You are liable for any infringement of copyright. Refer to your licensing agreement for more information.

The Online TV Program Guide video files cannot be redistributed. The Visual Basic 2.0 sample code can be reused as part of a new application but cannot be redistributed in its current form. The AVI installation program and AVI run time can be redistributed.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft[®]

