VBVoice - getting started

Other Topics

Included Demo Programs

LESSON1 - a simple application demostrating voice menus, playing greetings, and simulated dialling. (Requires voice prompts to be recorded). You can step through the creation of this application by reading TUT1.WRI

FAX - a application using WINFAX to send a fax.

INVDEMO - a sample database access system. Since this app uses databases, you must put VB into run mode before testing.

VMDEMO - a sample voice mail system. This app also uses databases for subsriber data (name , password etc). The prompts required for this demo are not supplied with the demo disk. They are available from the BBS on 613 839 0034. Download the file VMDMVAP.ZIP

USERDEMO - samples of how to create custom greetings using code, and also how to call the voice driver directly. Since the voice driver is not supplied with this demo disk, you will not be able to run this part of the demo.

Tutorials

For complete tutorials, see TUT1.WRI and TUT2.WRI.

TUT1.WRI steps you through making a simple voice application. (Lesson1).

TUT2.WRI is a more complex application involving multiple database searching.

Basic Concepts

Call Flow diagrams

Types of VBVoice controls

Characteristics of VBVoice controls

Greetings

Common Properties and Events

Call Flow diagrams

VBVoice uses Visual Basic forms to display and format the controls at design time. Although Visual Basic forms are intended to be graphical objects displayed to the user, VBVoice uses them primarily during the design phase as a container for controls. Once the application is complete and tested, this form will normally be hidden from the user.

A VBVoice system consists of one or more Visual Basic forms containing call flow diagrams. A call flow diagram consists of a group of VBVoice controls connected together graphically. These diagrams provide a graphical description of your system, and are used by VBVoice to create the system. Each VBVoice control performs a specific task related to the voice system. The Phone control, for instance, knows how to answer incoming calls, and how to initiate calls. When a control has performed its task, it passes the call to the next control in the tree. There may be several different routes possible, which are chosen based on the callers interaction with the system.
At the head of every call flow tree will be one or more Phone controls, each of which represents a telephone line.

This form answers the phone and during daytime it plays a greeting and takes a message, otherwise plays a greeting only
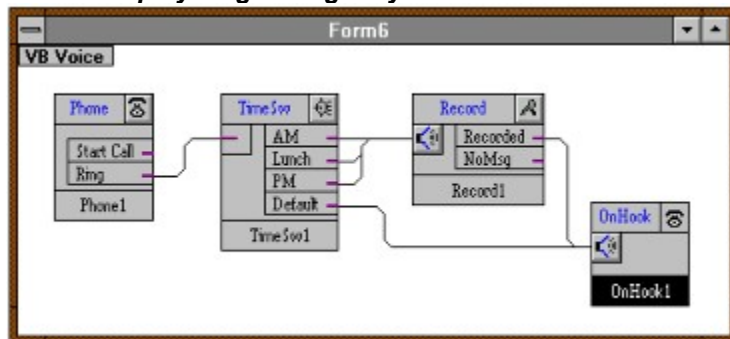
A telephone call can start in two ways: either incoming rings are received, or the Phone control will initiate outgoing calls. In the Phone control there are three output decisions, or nodes, called Ring, StartCall and Stopped. Incoming calls are routed to the Ring node and calls initiated by the Phone control are routed to the StartCall node. .
Either way the Phone control passes the call to the next control in the system, The call has then started to traverse the call flow diagram.
The next control in the call flow   diagram might be a GetDigits control. This control can play a greeting, and wait for the caller to enter a certain number or pattern of digits. The call is routed to the next control based on the digits entered. The call continues in this manner until it reaches an OnHook control, or the caller invokes invalid digit or time-out handling. Once the call is completed, control of the telephone line returns to the Phone control that owns the line, which will again wait for another ring or start the next call.

To create a system, VBVoice controls must be added to a form and connected together to form a call flow diagram. Each control has characteristics, or properties, that must be set.

VBVoice controls can use data obtained from controls earlier in the chain using Control Property Substitution .

Types of VBVoice controls

There are three types of VBVoice control:

- **Phone control**- The Phone control is a special control that 'owns' a specific telephone line, or voice channel. Each control can only own one channel at a time. Thus, if you have a four channel system, your system will normally have four Phone controls, one for each channel. Channels are normally allocated at design time, but can be allocated at run time if required. It can answer an incoming call, and can start a new call by going offhook.

- **Voice processing controls** - these controls interact with the caller or phone system - for instance playing greetings to the caller, receiving digits and recording messages. Controls in this category are GetDigits, Record, PlayMsgs, Dial, and OnHook.

- **Data and routing controls**- these controls do not interact with the voice system but can use data collected by voice processing controls - for example a DataFind control can look up a database record using digits entered by the caller and captured by a GetDigits control. The contents of a field in this database record can then be used by other voice processing controls to select greetings, direct the call, check a password or choose a transfer extension. Other controls in this category are Count, IniSw, DataGet, DataFind, DataNew, TimeSw, InConn and OutConn.

Characteristics of VBVoice controls

See Also <u>Changing control properties, greetings and names</u>

Control names

All Visual Basic controls have names. In Visual Basic, names are limited to 32 characters. VBVoice names should be shorter than this, since the size of the control will expand to allow all the name to be printed on the control. Control names are used to identify Visual Basic code procedures and to allow VBVoice controls to refer to each other to access data and variables.

Control Properties

All Visual Basic controls have properties, which are variables that can be set at design time and/or can be read and written at run time by Visual Basic code. VBVoice does not need the Properties window for setting parameters since all properties are accessible via the setup. You can save screen space by not displaying the properties window when working on VBVoice controls, by choosing the Properties menu item in the Visual Basic Window menu. See also <u>Control Property Substitution</u>

Greetings

VBVoice controls that interact directly with a caller usually have an initial greeting that is played when the call is transferred to that control. A VBVoice greeting is made up of one or more voice phrases that are played in sequence. The play usually terminates when the caller presses a key. Some controls have other greetings that are played to prompt the caller for more input, or to inform of errors, etc. See also <u>Greetings</u>

Connections

Most VBVoice controls have one or two input nodes on the left hand side, and one or more output nodes on its right hand side. A call enters the control via an input node, and when a control has completed processing of a call, it transfers the call to a control connected via one of the output nodes. The node used often depends on the result of its voice processing and is used to select the next appropriate action. Nodes are graphically connected on the screen to indicate the call flow either using drawn lines or via named connections when a graphical connection is not suitable - for instance when the connection is an error path that is not part of the main call flow, or when the diagram moves to a new form. See also <u>Connecting Controls</u>

Multi-form systems

When adding more than one form to a system, you must ensure that all VBVoice forms are loaded. Visual Basic defines one form or procedure at the startup form. This is defined in the Options / Project dialog. In your startup form, you should add some code in the Form.Load procedure to load all your VBVoice forms. To load the forms, use a Load statement, or set the forms WindowState or Visible properties. Normally you will want to display the forms while testing, but have them hidden when creating your .exe. Use a command line option to hide the forms in the run environment, but not in the VB environment. You can set command line options in Visual Basic environment using the Options... Project dialog. To hide a form without unloading it, set the forms Visible property to FALSE.
See the Form.Load procedure for the ADMIN form in the VMDEMO example.

See also Connecting to other forms

The Design Environment

Interacting with VBVoice controls

Changing control properties, greetings and names

Connecting controls

Greetings

VBVoice Main Window

Accessing control properties, greetings and names

Each VBVoice control has a set of properties associated with it. These properties are used to configure the control. They can be changed using the dialogs, and some can be changed at runtime by Visual Basic code

Click on the Setup button to set up the main parameters for the control.

Click on the speaker button to see the main entry greeting for a control. Release the mouse button over the speaker icon to change it.
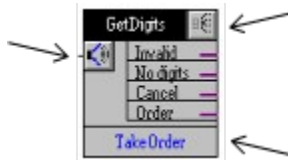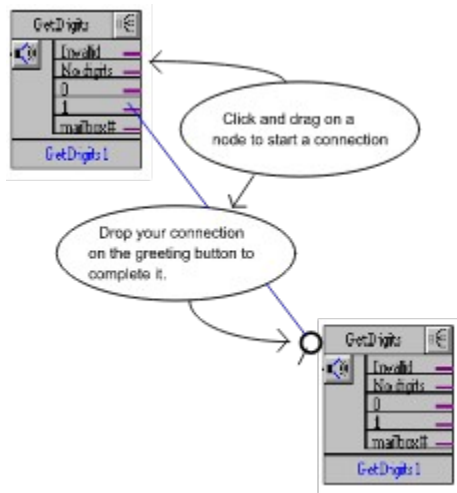

Double-click on the control name to change it.

The main greeting is played when the previous control has completed its operation and passes the call on the next. Some controls will replay this greeting when the caller enters invalid, or no digits.

Connecting controls

See Also <u>Connecting to other forms</u>

Using line connections

To connect a VBVoice control to another, click on an output node and drag to the input node of the control that will perform the next voice action, and then release. VBVoice will draw a 'rubber-band' as you drag, and when you release, it will draw the final connection.

Using named connections:

To make a named connection, click on the output node of the control and instead of dropping onto a control, drop onto a blank part of the form. A dialog will pop up showing all the currently available controls on this form. If a connection is already made from this node, it will be highlighted. Note forms must be displayed (either normal size or minimized) before their controls become available in this dialog. Use the Connect button to make a named connection, or the Disconnect button to remove an existing connection without replacing it.
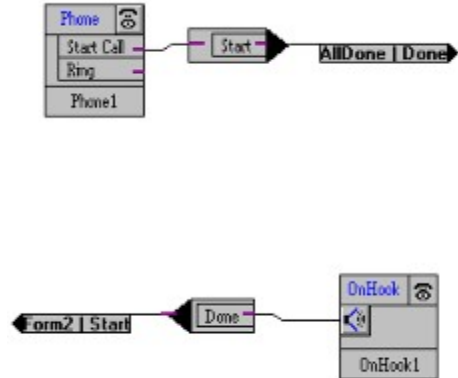

Deleting connections:

To delete a connection from one control to another, without replacing it with another, repeat the actions for creating a named connection to get the Connection dialog, and then click Disconnect.

Connecting to other forms

:

To make a connection to another form, you must use an OutConn on the source form and an InConn control at the destination.

Place an InConn control on the destination form and connect it using a line or named connection to the destination control. Now move to the source form and place an OutConn control and connect it to the source control. Name the InConn and OutConn controls to meaningful names (it helps later on). Now bring up the connection dialog as for a normal named connection by clicking and dragging on the OutConn control.

This dialog only shows InConn controls. The dialog has a form list as well as a control list, so you can list just the controls on a particular form by clicking on the form name.

Note:

Forms must be displayed (normal size or minimized) before their controls become available on the dialog. Use the Connect button to make a named connection, or Disconnect button to remove an existing connection without replacing it.

Greetings

Most controls have a built-in greeting. Each greeting consists of a list of phrases to play. There can be as many phrases as you want in a greeting. The phrases are concatenated together seamlessly to create complete sentences. You can also specify phrases using VB code to create <u>custom greetings</u> .

Phrase Types
Each phrase can be one of three basic types

a phrase from a <u>VAP file</u> .
Phrases can be selected from the vbvoice.vap or voicmail.vap files provided, or from a phrase that you record yourself.

a <u>voice file</u> .
A voice file contains compressed ADPCM data and is played in its entirety.

a <u>system phrase</u> .
There are system phrase types to say money, times, numbers, etc., and also a type called VBPhrase. Most system phrases have a parameter field, which is a text string set at design time. The string can contain control names. (See <u>Property Substitution</u>). Example: if the parameter field is BOX%MailBox%.VOX, and the MailBox control has collected the digits 123, VBVoice will translate the string to BOX123.VOX. Use database validation   to check the number first, to avoid phrase errors when the caller selects invalid digits - or handle the error in the error event.
A special type of system phrase is a VB Phrase -which allows you to add phrases to a predefined greeting at runtime. See   <u>custom greetings</u>

Listening to your greetings
VBVoice can play and record greetings using either a Windows sound card or a voice card. The demo system can only use a sound card.

VAP Phrases

VAP phrase files are files that contain many phrases. Each phrase has a text script associated with it. Using phrase files simplifies script generation and management by allowing the full script to be stored with the phrase. In addition, it reduces the number of files in use at any time, which increases speed of operation and reduces the use of system resources.

System Phrases

The System Phrase dialog is used to add a System phrase to the greeting. System phrases are defined to say money, times, dates, numbers and other system phrases. Most system phrases require a parameter, which can contain a control name. (See Property Substitution).

Date	Parameter: a date in the format dd/mm/yy.
Digits	VBVoice says a number or string as a series of digits and letters.
FileDate	As above, but VBVoice says the last modified date of the file.
FileSize	Parameter: a file path. VBVoice will say the duration in seconds of the file using the normal voice file sampling rate and compression type. The path can contain control names. The file path should be a fully qualified path name.
FileSpec	Parameter: the name of a voice file in the VBVoice directory, or a full path to a voice file. Example: if the parameter field is BOX%MAILBOX%.VOX, and the MailBox control has collected the digits 123, VBVoice will attempt to play a file BOX123.VOX from the default VBVoice directory.
FileTime	As above, but VBVoice says the last modified time of the file.
Initial Greeting	This phrase is intended to be used as the first phrase in the main voice system prompt. It says Good Morning, Good Afternoon, or Good Evening based on time of day. Before noon, it says good morning, from noon to 5 PM, it says good afternoon, and from 5 PM to midnight it says good evening. The phrases used are in VBASE.VAP and can be changed to suit your requirements. The position of the phrases in the file should not be changed.
Money	VBVoice translates a number into an amount, e.g. 1234 is "twelve dollars and 34 cents".
Number	VBVoice says the numeric value of the parameter, i.e. the string 1563 would be 'one thousand, five hundred and sixty three'. This phrase type can also handle decimal numbers, e.g. 123.45 would say one hundred and twenty three point four five.
Number(Ordinal)	As Number, but says first, second etc. Decimal points are not allowed in this number.
NumberShort	As Number, but numbers greater than 9 are said as "more than nine".
NumFiles	Parameter: a file path that contains a drive, directory and wild cards e.g. MAILBOX*.vox. VBVoice will say the number of files found that match the file specification.
Rotator	Parameter: VAP file name. A rotator is a phrase that says one of a number of phrases in rotation. The phrase to be played is selected from one of the phrases in the VAP file. Each phrase is played in sequence, regardless of the channel upon which the phrase is played.This is useful when you are playing advertising messages, and want to give each phrase an equal amount of exposure.
Time	Parameter: a time in the format hh:mm.
VAP Phrase by Index	Parameters: a VAP file name, and an index number. The index number selects the phrase to play. Phrases in the VAP file are numbered from 1. Both the file name and phrase index can contain control names.
VAP Phrase by Name	Parameters: a VAP file name, and a phrase name (script). The filename and phrase name can contain control names to allow phrases to be selected at runtime by control properties.
VBPhrase	This is a phrase type that allows VB code to create a phrase at runtime. A name is

associated with the phrase so that each phrase can be identified. When a control attempts to play a greeting containing a VBPhrase, a PlayRequest event will be generated. VB code can then set the file or phrase to play. There can be as many VBPhrases as required in a greeting. Phrases can be created using the DLL functions described below. There is no limit on the number of phrases in a greeting (See PlayRequest event, page I-31).

A PhraseError event is generated if an error occurs during phrase processing, such as the specified file does not exist, or an invalid control name has been specified. See Errors page I-33.

Voice Files

The file dialog allows you to add a compressed ADPCM voice file to play in your greeting. If you wish to dynamically select a file to play based on previous digits or database entries, use the System phrase type File Spec instead. The File dialog is a standard Windows file selection dialog, except VBVoice adds a button Play. This will play the selected file before you add it to the greeting.

Custom greetings

To create a single phrase that is configured at run time, use a VBPhrase.
The System Phrase **VB Phrase** allows your code to select a phrase to play in addition to the. When the system encounters a VBPhrase, it will call the PlayRequest event procedure defined in the control and passes in the name of that VBPhrase. Your event procedure can then create a phrase object using the provided DLL functions,   which is returned by the procedure. The control will then play the phrase. See also

 For more complex custom greetings, use the custom greeting capability which allows you to assemble complete greetings using code..
This consists of some DLL functions, and a parameter in the Enter event.

The functions are:
vbv_create_greeting
vbv_add_phrase_to_greeting
vbv_clear_greeting
vbv_destroy_greeting

See DLL Functions in the main VBVoice help file for more information on each function.

See   the VBVoice main help file for details on System Phrase Types.

Control Property Substitution

VBVoice has a powerful mechanism to control the system based upon data collected from the caller, using control property data values.

Default properties
Most VBVoice controls have a default property value. This value is available for use by other controls, in greetings and other setup fields. For instance, the GetDigits default property is the Digits property, containing the digits collected from the caller. This collected digit string could be used in a DataFind control to select the database record to be found, or to select the file to be played as a greeting in another GetDigits control.
Note that VBVoice uses the control name to identify properties. Therefore, no two control names should have the same name, even if on different forms. VBVoice will check for this condition during the system check.

Syntax
To tell VBVoice that you want to substitute a control value, surround the name of the control with % marks - for example %GetDigits%. At run time, the %xxx% string will be replaced with the actual value of the default property. If you copy the name from the Controls List dialog, and paste it into the dialog, the % marks will be added for you.

Adding a Control Name
To find a control name, use the Find Control button to move to the Controls List dialog. Copy a control name to the clipboard and then paste it into the parameter field by selecting it and typing Shift+Insert. At run time the actual value of the controls default property will be substituted for the control name in the parameter string.
See System Phrases in the main help file for more details on the system phrase types.

Ways to use Control Property Substitution

Say an number entered by the caller
If you have collected some digits from a caller, and with to play them for confirmation, use the SayNumber phrase.
Say you have used a GetDigits control called GetMailBox to collect the digits. To play back the numbers that the user has entered, use the system phrase SayDigits or SayNumber in another control, with the parameter %GetMailbox%.

Data verification
To verify data entered by the caller, create a database containing all the valid numbers or digits.
To add verification to the example above, add a data control that is connected to your database via the DatabaseName and RecordSource properties. Add a DataFind control, and in the setup dialog, set the Search in database field to the name of your data control, set the For...   to the %GetPartNum% and set the Field entry to the name of the data field containing your valid data. Then, if the digits collected are in the database, the call will proceed out of node "Found". If there is no match, the call will continue out of the node "Not Found". In this case, you may want to give the caller another try, just as if the digits had been found invalid by the GetDigits control itself. To do this, connect the NotFound output back to the Err input on the GetDigits. This will increment the error count in the GetDigits control, play the error greeting followed by the main greeting and give the caller another change to enter a valid number. After the maximum number of retries have been made, the call will exit via the Invalid output in GetDigits.

Look up numeric data and say it
VBVoice can also say numbers from the database. For instance, to add the example above, your database containing valid numbers can have another field Quantity.
Use a DataGet control called GetQuantity to access the field by connecting it to the Found output of the DataFind control, and setting the Get Data from Field entry to the name Quantity. (It should appear in the

list provided). No conditions are required in this example. Now in a subsequent voice control, use a greeting containing a phrase:
System Phrase Say Number %Quantity%

Look up other data and say that too

Your database could also have another field Description containing a list of filenames, each of which is a spoken description of the item. Add another DataGet control with the Get Data from Field entry to the name Description. Now

VBVoice can also say data from the database. For instance, to add the example above, your database containing valid numbers can have another field containing the filename of a prompt describing the record. To say the description, use a greeting containing a phrase:

VAP Phrase: "You have selected"
System Phrase FileSpec<%Description %
to say a description of the item the caller has selected.

Another way of solving this problem is to use a VB Phrase.
VBVoice will call the PlayRequest event procedure whenever it needs to play this phrase. Your code can read the current properties of all VBVoice controls, as well as your own data, and decide which file or phrase to play.

Look up data using letters on the keypad

You can search databases using letter substitution for the numerals on the keypad, and cycle through all the entries found - useful for directories in voice mail applications. Set the Alpha digits field in the DataFind control to do a search based on phone key to alphabetic translation.

VBVoice Main Window
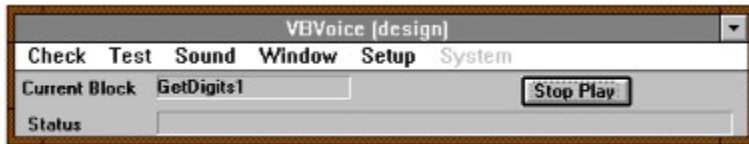
Check Menu

Use the Check Control, Check Form, and Check System commands to check your system. Even if you have only one form in your system, Check System performs some extra checks over Check Form. If you get errors, double click on the error message to highlight the control in error

Check Control

Check control will perform an error check on the currently selected control. If a log window is not visible, one will be created to display the results of the check. If an error is found, you can double-click on the error message to jump the control in error. To get more information on the error, select the error line and press F1

Check Form

This command will perform a Check Control command on all the controls on the current form.

Check System

This command will perform a Check Form on all forms currently loaded. Note that forms in the project that are not displayed, either normal size or minimized, will not be included in the check.

Test Menu

This command will start VBVoice in Test Mode. The test will not start until you use the Start Test command. See Test Mode

Sound Menu

Use Voice card
Use Sound card

These commands allow you to choose which sound device to use for listening to greetings, and for testing the system. In the Demo system, the Use Voice card option is always disabled.

Voicecard Options

This command shows the Voice Card options dialog. This allows you to configure how you will be using your voice card during testing.

Window Menu

Line Status

This command is enabled when VBVoice has started the voice system. It will display the Line Status window, if not displayed

Log Windows

This command allows you to select which channels you want to monitor as your system is running. The dialog shows which channels are currently being monitored with a check mark in the box for that channel. When you click OK in the dialog, log windows are opened or closed to reflect the status you entered in the dialog. Showing too many log windows may slow down your system especially when monitoring all events. .

Setup Menu

Directories

This dialog allows you to change the path to ANNOUNCE.EXE, the voice editor, and the default directory that VBVoice uses to find phrase files.

PBX

This dialog allows you to change the way that VBVoice interacts with the phone system to perform

operations such as transfer, put on hold, etc.

Error Handlers

Use this dialog to define the greeting files that will be played when the default invalid digit or time-out handlers are invoked. See GetDigits, PlayMsgs, Record and Count for details on when these error handlers are used.

Startup Files

This dialog allows you to specify which voice and phrase files will be opened when the voice system is started. Opening files at start-up time speeds up execution by eliminating the time required to open and close the file for each play operation. However, file handles are a limited resource in the system, so files that are only used occasionally should be opened when required.

Before using this dialog, load all your voice forms so that VBVoice can scan them all to produce an accurate list of all voice files in use.

System Menu

Start System

When Visual Basic is in run mode, the Start System menu item is enabled. Start System performs the same function as the DLL function vbv_start_system(). It checks all the controls, loads the voice card driver and start voice operations. It also displays the Line Status window. This command requires a voice card driver.

Stop System

Once the system is started, you can stop voice operations with Stop System. Normally VBVoice will wait for all lines to clear before terminating. The following events will occur:

- The Line Status window will appear and will mark all idle lines as waiting for termination.
- It will then show the Wait for Termination dialog while waiting for lines to clear.

If you want to terminate immediately, use the Stop All Lines button in the dialog. All lines will be cleared immediately, and the system shut down. Otherwise VBVoice shut down the voice system when all lines have been released.

Forcing a system stop...

If no valid events are received from the voice card driver, it may not be able to complete an orderly shutdown. If this happens, close the dialog using the dialog System menu (use the button at the top left of the window) and select Close to force termination. Some greeting files may be left open, so the application should be restarted.

Log File Setup

This command will allow you to specify the level of detail for the log file. See Event Logs.

Authorization

This command allows you to enter an authorization code to upgrade your system to a full system.

Test Mode

Test mode allows you to start a test of your system at any point in the call flow diagram. Show the Test Mode dialog by selecting the Test menu item in the status window. The control that will start the test is highlighted (white text on black) and is shown in the test dialog name. You can change the start control by clicking on the required control, or it can be selected via the Start Control**...** button.

Testing Subsystems and Individual Controls

Visual Basic run mode vs design mode

Using the Sound Card in Test mode

VBVoice Test Dialog

Testing Subsystems and Individual Controls

Test mode is useful for testing a subsystem without having to worry about how the call is normally routed to the subsystem. In complex multi-level menus, it is convenient to be able to start at any point in the system, however, consideration must be given to the interactions between controls in your system. Consider the scenario where a Record control uses control name substitution from a previous GetDigits control to create a filename based on the entered mailbox digits to direct the file to the correct mailbox directory. If test mode is started at the Record control, the GetDigits control will not have collected any digits. VBVoice detects this condition, and will present a dialog requesting that you fill in the value that the GetDigits control would have supplied. Again, you can enter an invalid value to check all error conditions.

Starting a test

Start a test by selecting the control where you want to start the test, either by clicking on it, or choosing Start In... and selecting a control name. Then select the Start command to start the test. If you are using a voice card, which is onhook, you may be prompted to dial in to start the test.

When the test starts, the test log window will be shown. This window monitors events as they occur in the system. If an error occurs, it will be logged in this window, and the test will stop. To get more information on the error, select the error line and press F1.

If there is a validation error in the control, the test will not start and an error message will appear in the test log.

Control Checks

Each control will be checked for setup errors before a call is passed to it, so it is not necessary to have the complete system error-free before starting a test. However you may find it useful to run the Check Control command before starting test mode.

Call Flow

You can see you how your call is progressing during the test. The call always starts in the control with the highlighted name. If the call moves to another control, the originating node and the line connecting the two are highlighted. If an error occurs, or the call is terminated with the Stop button, the control name of the last control is highlighted in dark red, allowing you to zero in on error locations quickly.

Timeouts and Caller hangup

The test dialog provides Timeout and Hangup buttons which allow you to simulate a time-out while waiting for digits, or a caller hangup at any time.

Outdialing and Call Progress

When in test mode, VBVoice will dial but will not perform transfers or call progress detection. It allows you the select the call progress condition that you want to test from a dialog, so you can test all control paths without having to recreate (and wait for) ring-no-answer, busy, etc..

Run mode vs Design mode

You can initiate a test in both visual Basic run mode and design mode. Testing in design mode is quick and easy, but does have some limitations that you should be aware of:

Testing in Visual Basic design mode - limitations

Database access:
Database access is not available in design mode. If the call arrives at a DataFind control or DataNew control in Test mode, and Visual Basic is in design mode, the test will terminate.

Visual Basic code:
If your design relies on Visual Basic code, in custom VB phrase events, Enter events   or elsewhere, this code will not be run while in design mode.
.

Testing in Visual Basic run mode

Test mode can be invoked while Visual Basic is in run mode. In this case, database access is available and Visual Basic event procedures will be called.

You must ensure that Visual Basic loads the required forms at startup. This is a requirement for normal operation as well as for test mode. If only one form is to be tested, it can be made the startup form in the project options dialog (in the Options menu). If more than one form is required, the form load procedure in the startup form or module must load all other required forms. You can load the form as an icon by setting the WindowState property to 1.

Visual Basic debugging:
Test mode does not interact with Visual Basic debug mode, as it does when running using Start System. If Visual Basic switches to break mode due to an error in your code, Test mode will terminate.

Using the Sound Card in Test mode

A sound card can be used for playing greetings in design mode and for testing your system. VBVoice simulates actions relating to the telephone system such as dialing.

Dialing

During dialing operations, VBVoice will say the number dialed using the sound card. If call supervision is being performed, VBVoice will show a dialog allowing you to select the call progress type to be simulated - call answer, busy, no dialtone etc. This allows you to test the system without having to wait through 30 seconds of ringing to test the no-answer handling.

Dialed digits

Dialed digits can be entered at any time using the dial pad in the Test dialog as if you were the caller.

Recording

VBVoice uses prerecorded files as the source of data when making recordings. This saves time, and eliminates the need for a microphone. VBVoice shows a file select dialog when a record operation begins. The VOX file you select is copied into the file that would have been recorded.

VBVoice Test Dialog

When VBVoice test mode is started, access to the Visual Basic and VBVoice main menus is denied. Terminate test mode with the Cancel button to return to normal operation.

The VBVoice Test Dialog (when using a Sound Card)

Start In ...
This command allows you to select the control at which you want to start the test. You can also do this by clicking in the control.

Start / Stop
Use this command to start the test. Once the test is active, you will not be able to make any design changes until the test is stopped.

Timeout
The time-out command allows you to simulate a time-out condition when performing a delay operation, or a no digits or silence time-out when performing a get digits operation. This allows you to quickly test time-outs without having to wait for the time-out to actually expire.

HungUp
The HungUp command simulates the caller hanging up the phone to terminate the call when using a sound card, and avoids the need to drop the line when using a voice card.

Pause/Resume (Sound card only)
When using a sound card, you can pause play, get digits and delay operations until you are ready to resume.

Cancel
This command will terminate test mode and will return the system to normal operation.

Dial Pad (Sound card only)
The dial pad is used to enter digits from the caller when testing using a sound card.

Common Properties and Events

This chapter describes events and properties that are common to almost all VBVoice controls.

Accessing Properties

Most properties of VBVoice controls are array properties. Using arrays allows VBVoice to handle multiple channels concurrently within the same application. Normally there is a copy of each property for each channel, therefore to access a property you must specify the channel in question. All VBVoice events provide a channel parameter which can be used to index the properties. The syntax required to index arrays in Visual Basic is *variable_name(index).* For example, to set the GotoNode property in the Enter event for control User1, use this code:

```
Sub User1_Enter(channel as Integer)
User1.GotoNode(channel)= 0
End Sub
```

Errors can occur while setting properties in VBVoice controls. See Error handling for more information

GotoControl, GotoNode

These properties can be set in most controls by code in the Enter event. Setting one of these properties will bypass the normal action of the control and route the call directly to the specified control.

GotoNode

Setting this property to a valid node number in the control where the event occurs will route the call to the control connected to that node. Node numbers start at 0, numbered from the top.
For instance, if GotoNode is set to 0, the call is passed to the control connected to the first (top) node.
The transfer takes place as soon as the Enter event code is completed.
An error event will be generated if the node number is invalid, or not connected.

Example

To transfer to the control connected to the fifth output from the top, node 4:
```
GetDigits1.GotoNode(channel) = 4
```

GotoControl

Setting this property to a valid control name will route the call to that control. The destination control does not need to be connected to the current control, and does not need to be on the same form. If GotoControl is set to "MailBox", the call is passed to the control named MailBox.
If the destination control is part of an array of controls, the control name used should be of the form ControlName(index), e.g. GetDigits(23).
An error event will be generated if the control name is not found. The transfer takes place as soon as the Enter event code is completed.

Example

Go to #5 in the array of User1 controls:
```
GetDigits1.GotoControl(channel) = User1(5)
```

Event handlers

Below is a description of the events that occur in almost all controls. Refer to the main VBVoice help file for a description of events specific to each control

Disconnect event

This event procedure is called when a caller hangs up. It can occur in any control that does voice processing. This event allows clean-up on a per-control basis. After this event occurs, a Disconnect event occurs in the Phone control that initiated this call. This hierarchy of events allows localized clean-up procedures and/or global clean-up as appropriate. The global Disconnect event occurs regardless of whether the caller hangs up first or the system hangs up the call. The local Disconnect event only occurs when the caller hangs up.

There are three possible reasons for hangup: caller hangup detected, default error handling, or an onhook control. The event occurs after the line has been taken on-hook.

> *Sub xx_Disconnect (Channel as Integer, Reason as Integer)*

Possible disconnect reasons:

Const CONTROLHANGUP = 0	the call terminated at an Onhook control
Const SYSERRORHANGUP = 1	the call terminated due to a system error in a control
Const CALLERHANGUP = 2	the call terminated due to the caller hang-up indication from the telephone switch
Const INVALIDHANGUP = 3	the call terminated due to default invalid digits or no digits timeout handling
Const SYSSTOPPING=4	The system is stopping due to a Stop All Lines command

Enter, EnterB event

One of these events occur when a call enters a control, depending on the input by which the call arrives. The EnterB event occurs in controls that have 2 input nodes, when the call enters via the secondary input. Enter events give VB code an opportunity to change some of the characteristics of the control by setting its properties on a call-by-call basis, or to bypass the control altogether by setting the GotoNode or GotoControl properties.

The Enter event has an additional parameter, Greeting. This parameter can be set to a greeting object, created with one of the DLL greeting functions. Your code can create greetings composed at runtime and allows complete flexibility in composition of greetings. See Custom Greetings, page 14.

> *Sub xx_Enter(Channel as Integer, Greeting as Long)*
> *Sub xx_EnterB(Channel as Integer)*

Exit event

This event occurs when a call leaves a control, and before the Enter event occurs in the destination control. The node parameter is the node number which indicates the destination control. Nodes are numbered from 0.

> *Sub xx_Exit (Channel As Integer, Node As Integer)*

PlayRequest event

This event occurs when a greeting contains a VBPhrase. Each VB Phrase is named so that it can be identified. In the PlayRequest event procedure, your VB code should create a phrase object using the DLL functions, and return it in the PhraseObject parameter, or return 0 to skip the phrase.

> *Sub PlayRequest(Channel as Integer, PhraseName as String, PhraseObject as Long)*

PhraseError Event

VoiceError Event

> *See next chapter.*

Using databases

Creating a database
You do not need a special application to create simple databases. Visual Basic provides a tool, DATAMGR.EXE which can create and update databases. While DataMgr is limited, it does provide a quick and simple way to view and create databases.

Setting up the Microsoft data control.
At first sight the Visual Basic data control appears to have a daunting array of properties, but it requires only 2 properties to connect it to a database. These are DatabaseName and RecordSource.

The DatabaseName property defines the file containing the database. This file should be the Microsoft Access database file, created with Access or with the DataMgr utility. The DATAMGR.EXE is located in your VB directory. If using Access, you can also set up attached databases, so that VBVoice can access remote databases from a database server.

The RecordSource property defines the recordset within that database that the data control will access. Although it is possible to use SQL statements in the RecordSource property, for most applications a table name will suffice.
If you are not writing to your database, you can improve performance by setting the ReadOnly property to TRUE.

A sample SQL statement

        SELECT Titles.Title.Dept, Author FROM Titles, Authors WHERE Titles.AU_ID = Authors.AU_ID

When you set the RecordSource to a table name, the data control will be able to access all the records in that table. If you will not be updating the database, you can set the ReadOnly property to True.

Accessing the data
You can add some text boxes to your form, and link them to fields in the database. The text boxes will display the contents of the current record in the database, and will also allow you to change the data in the database.
To link a text box to a database, perform the following operations:
- Add a text box to your form. You can use a label also, but this will not allow you to change the data.
- Set the DataSource property to the name of the data control to which you want to link.
- Set the DataField property to the field name in the table.

Now, when you start your program running, the text box will show the contents of the first record in the database.
To scan through the database, use the arrows on the data control.

Changing the data
To change the database contents, put some new text in the text box. The database will be updated when you move the data control to a new record.
Overriding the DataFind search algorithm
The standard database search algorithm provided by DataFind can be slow in large databases, and does not support SQL searches at runtime. You can select the RecordSet at startup using the RecordSource property, however this should not be changed at runtime (unless you are sure that there are no channels that have bookmarks defined). You can override the standard search algorithm to improve performance or to add SQL search statements based on control properties such as collected digits.

Improving database performance
The DataFind control searches the database using a MoveNext / GetData / Compare algorithm. This can be inefficient when searching large databases. To overcome this, override the standard search algorithm using the code described below.

This code overrides the normal action of the DataFind control to use the commands FindFirst and FindNext, while still supporting the other data controls.

In addition, the file PERFORM.TXT in your VB directory contains information relevant to improving database speed from within Visual Basic. In particular, you can improve performance by setting the database ReadOnly using the   ReadOnly property of the data control. You can also increase the data control cache size with   the maxbuffersize entry in VB.INI.

Also note that having text controls bound to the data control slows down searches considerably. These have been added to the Vmdemo and Invdemo forms to show you the current contents of the database only. If you wish to display the current contents of the database on one of your forms, use another data control, or turn off the link from the text control to the data control during the search.

Using SQL statements as search criteria

If you wish to add more complex search criteria to your database searches than that provided by the DataFind control, you can override the standard search mechanism to incorporate SQL statements, as described below. The string mycrit in the example below can be any SQL string, and can include your own variables and VBVoice control properties.

Example

In the form declarations, declare an array of strings to hold the position of the DataFind for each channel.

```
Dim bookmarks(1 to MAXCHANNELS) as String
```

In the Enter event, add the code below. This code sets up a search criteria string, and uses the FindFirst method to find the first record. If a record is found, the GotoControl property is used to transfer the call to the control connected to the Found output, otherwise the call is transferred to the Not Found output. Note error handling has been omitted.

Sub DataFind1_Enter (Channel As Integer, Greeting as Long)
```
        Dim mycrit As String

        move to the beginning of the recordset
        DataControl.Recordset.MoveFirst

        'do the find operation
        DataControl.Recordset.FindFirst mycrit

        ' remember the position for the EnterB event
        bookmarks(channel) = DataControl.Recordset.Bookmark

         this duplicates the found / not found branching
         setting the gotonode makes the call exit immediately, overriding the
         normal action of the datafind
        If DataControl.Recordset.NoMatch Then
        DataFind1.GotoNode(channel) = 2
        Else
        DataFind1.GotoNode(channel) = 0
        End If
        End Sub
```

The EnterB event is similar, but uses FindNext instead of FindFirst, and if no match is found, it transfers to the Not Found exit, rather than the None Found exit.

Sub DataFind1_EnterB (channel As Integer)
```
        Dim mycrit As String

        goto bookmark for this channel
        DataControl.Recordset.Bookmark = bookmarks(channel)
```

set search criteria for Student_ID database field to equal digits received
mycrit = " StockNumber = '" + GetDigits1.Digits(channel) + "'"
DataControl.Recordset.FindNext mycrit

If DataControl.Recordset.NoMatch Then
DataFind1.GotoNode(channel) = 1
Else
DataFind1.GotoNode(channel) = 0
End If
End Sub

Multiple language support

VBVoice can support up to 12 languages simultaneously, if your system is authorized for the multi-language option. There are three basic requirements for multi-language support:

- each call can select its own language.
- the actual voice files to be played must be selected based on the current language.
- the system phrases for dates, times, numbers etc. must be created using the format required for the current language.

The phrase files for the default language (language 0) are kept in the VBV directory. Additional phrase files for each language are kept in directories LANG1, LANG2 etc. up to the number of languages in use. These directories must be subdirectories of the VBV directory.

Date and number formats are controlled by VBVOICE.INI settings. For instance, to set language 1 to be French and language 2 to be Spanish, use

[Languages]
Language1 = French
Language2 = Spanish

Contact PRONEXUS for availability of these and other languages. If there is no setting, the default format (English) is used.

Please note these important points:

- Each directory must contain a complete set of phrase files required for your application.
- Each language phrase file must have the phrases in the same order as the equivalent phrase file in the VBV directory.
- Only phrase files referenced without a path name will be substituted by the correct language file. If you reference a file directly using C:\VBV\VBVOICE.VAP, then this phrase file will be used regardless of the language.

Changing language

To change to another language for a particular call, use the DLL function *vbv_set_language*. This will not affect any other calls. Pass in the channel number that this call is operating on (available in the Enter and Exit events), and the language number. This can 0 (default language), up to 11.

Example:

vbv_set_language channel, 1

Each call starts with the default language, language 0. All voice files that do not have full path names will use the language setting to select the voice file location. If a phrase is created using a full path name, the language setting is ignored.

Event Logs

VBVoice can log events to hard disk and to log windows during operation. The messages can be filtered depending on message type.

> ***Event types to monitor:***
> **errors**
> **calls** (the start and stop time for each call, and the system start and stop times)
> **call flow** (a call flow message is logged whenever a call leaves or enters a control)
> **information** messages (these are general information messages about the current action of each control)
> **invalid digits, caller time-outs** (these log the events that occur when a control detects a timeout or invalid digits from the caller.
> **driver events** (low level voice driver events)
> **driver functions** (low level voice driver function calls)
> **control states** (These messages reflect the internal state machines used by each control)

Help
To get help on a particular log entry, select the log entry and choose the Help command in the log window menu bar.

The VBVoice Log File
 A new log file is created every day to avoid the log file becoming unnecessarily large. The file name for the log files are of the form vbvdd-mm.vlg, where dd is the day, and mm the month. Log file size may still become a problem if all events are monitored to the log file. Normally calls, call flow and error events should only be logged to the log file. You can choose which event types to log using the LogFiles item in the Setup menu.
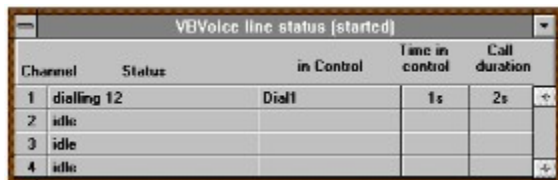
Channel log windows
VBVoice can show a log window for each channel in the system and maintains a log file of events. You can choose which types of events to display in a log window in the same way as to the log file. Each window is associated with one channel. You can select separately which events to monitor. In addition to showing events as they occur, log windows can extract information from the log files and display it in the same list.   Note that having many log windows open may slow down your system.
You can also save the contents of a log window to a text file.

Line status window
This window shows the status of each channel in a compact format, one line per channel.

The VBVoice Toolbox

Count	Maintains an internal counter, which can be reset or incremented. The call is transferred to a new control depending on the results of a comparison between the counter value and a preset limit. Useful for simple loops.
DataChange	Changes data in a database record, in one or more fields. The database and record are selected by a previous DataFind or DataNew control.
DataFind	Searches for the first or the next database field depending on whether the control is entered via the main input or the "Next" input. A wide range of data match conditions is available.
DataGet	Reads a database record from a previous DataFind control, stores the result in the property 'Result', and routes according to the value obtained.
DataNew	Adds a new, blank record to a database.
Delay	Implements a wait period. The caller can be put on hold, or a short voice clip can be played during the wait.
Dial	Dials some digits either during a call or to start a call. Can perform call supervision (answer, no answer, busy etc.) and has built-in support for PBX call transfer.
GetDigits	Plays a greeting, and waits for digits from the caller. Configurable exit conditions using wild characters, numeric characters and ranges. Built-in invalid digit and no digits handling.
Greeting	Plays a VBVoice greeting. Allows option digits for fast forward, rewind and pause.
InConn	Used for connection across forms. Other controls cannot be connected across forms.
IniSw	Searches for a setting in a Windows initialization file. Branches to another control depending on the value found.
OnHook	Plays a greeting, and hangs up the phone.
OutConn	Used for connection across forms. Other controls cannot be connected across forms.
Phone	This control will wait for incoming ringing on a channel, or it can start a call. After a preset number of rings, the channel is taken off hook and control is passed to the next control.
PlayMsgs	A high level control that accesses a database file containing a list of messages. Configurable options after play. Supports new, old and deleted messages. Updates database with new message status.
Record	Plays a greeting, records a file, and adds a new record to a database. At the end of the recording, allows the caller to choose options to re-record, delete, append to end of recording, or to save the message.

	TimeSw	Transfers the call to another control according to time of day and day of week.
	User	This control allows you to use VB code to create your own control. You have access to all the high-level VBVoice functions and the low-level hardware-specific driver functions.