Writing VOS Script 7

This chapter begins with a tutorial that teaches you how to create two programs using the VOS language. Next, it analyzes the code from one of the programs you create. Finally, it provides some commonly used telephony functions written in VOS. It contains the following sections:

- Starting the VOS Program Tutorial
- Examining the Echo Program
- Using Basic Telephony Functions

Starting the VOS Program Tutorial

Before beginning this tutorial, make sure Graphical VOS is installed on your machine correctly. You are also going to use SimPhone, which is automatically included in the CT ADE installation.

If you have questions or want more information on any of the VOS code you see during this tutorial, see the Graphical VOS online help. The online help also includes a description of each VOS function.

Creating a Project for Your Program

Complete the following steps to create a Graphical VOS project (projects manage the files needed to run your VOS script programs).

1. Start SimPhone and Graphical VOS from the Windows Start menu.
2. On the Welcome to Graphical VOS dialog, select Create a new project, then click OK to display the New dialog.

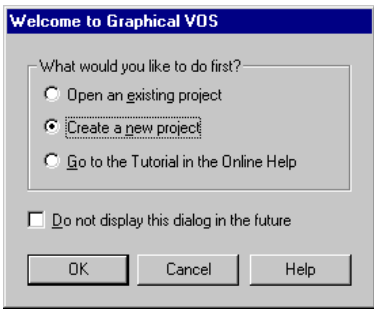


Figure 7-1 Welcome to Graphical VOS dialog

3. On the New dialog:
    a. Make sure that Project is the selected File Type.
    b. In the File name field, name your project **LanguageTutorial**.
    c. In the Location field, type the following path for the project:

```
C:\Program Files\Parity Software\Graphical
VOS\Tutorial\Language
```

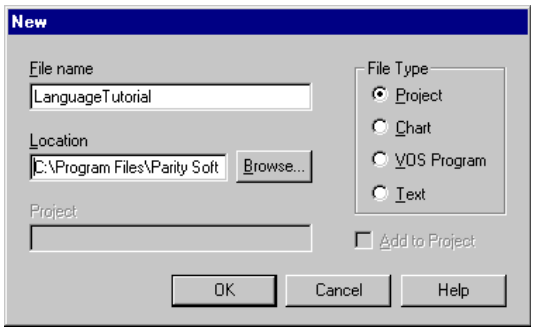


Figure 7-2 New dialog

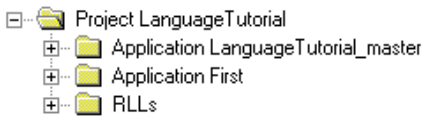
    d. Click OK, and Graphical VOS creates your project.

Adding a Program to the Project

Now it's time to add a VOS program to the project.

1. From the main menu in Graphical VOS, click File | New.
2. In the New dialog:
    a. Make sure that VOS Program is the selected File Type.
    b. In the File name field, name the program **First**.
    c. Make sure the Add to Project checkbox is checked.
    d. Click OK to create the new program.

    A folder called Application First is added to the project tree in the left window:



3. Double-click the Application First folder, and you'll see an icon labeled First.vs in the project tree.

The Editor window to the right is blank because this new file currently contains no text.

4. Type the following program into the Editor window:

```
program

    vid_write("I am a VOS program!");

    vid_write("Type any key to exit...");

    kb_get();

    vid_write("Exiting now.");

    exit(0);

endprogram
```

Compiling and Running the First Program

When you compile a VOS program, it gets converted from a form that you can understand (such as a source file, like First.vs) to a form that VOS can execute (such as a binary file—also called an executable file).

1. From the Graphical VOS main menu, choose Run | Compile First.vs.

   If you didn't make any typing errors, you'll see a message telling you that your program has been compiled:

```
Compiling...
first.vs
Compilation Complete
```

   The VOS Language Compile (VLC) created a binary file called First.vx. You can use Windows Explorer to see that the binary file has been created. If you see error messages from VLC, check your typing by carefully comparing the file you created with the source code provided.

   Next, run the First program:

2. From the main menu, choose Run | Run First.vs.

   Notice that Graphical VOS compiled your program again. Any time you run a program in Graphical VOS, it automatically compiles the program first. After compiling the program, Graphical VOS starts the VOS runtime engine. You can see the stopwatch icon ticking in the Windows system tray, and a new window called VOS Box opens (if you have previously disabled the VOS box, it doesn't open).

3. Right-click the VOS Box icon in the Windows system tray, and choose Restore.

The following message appears in the VOS Box:

```
I am a VOS program!
Type any key to exit...
```

4. Type any key, and the VOS Box window closes.

   If you didn't see that message, try looking at the VOS log file. From the main menu, choose View | Runtime Log. The VOS1.LOG file is created automatically by VOS. It displays a new message each time VOS is started, and each time a condition arises that may indicate an error in a program or hardware problem. If you don't find a message indicating the source of the trouble, it's time for your first tech support call. For more information, see the chapter titled "Technical Support" in this guide.

Reading the Source Code

You should find the source code to be mostly self-explanatory. For example, the word *program* starts a program, and the word *endprogram* marks the end of the program.

The vid_write function asks VOS to write the given string to the VOS Box. The kb_get function waits for a keystroke, and returns the character pressed, although we ignore the returned value in this example. Most VOS programmers use languages like Visual Basic or Visual C++ to develop interfaces for VOS programs so you probably won't use functions like vid_write and kb_get very often, but they're appropriately simple for an example like this.

The exit function stops the VOS runtime engine with the given exit code (which can be tested by the IF ERRORLEVEL command in batch files, etc.).

For more information about interpreting the source code, see the chapter titled, "VOS Language Concepts" in this guide or the Graphical VOS online help.

Creating a Second Call Processing Program

Now you're ready to add another VOS program to the project.

1. From the main menu in Graphical VOS, click File | New.
2. In the New dialog:
   a. Make sure that VOS Program is the selected File Type.
   b. In the File name field, name the program **Echo**.
   c. Make sure the Add to Project checkbox is checked.

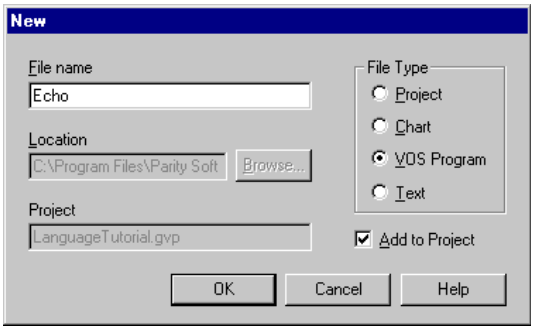


Figure 7-3 New dialog displays when you choose New from the Graphical VOS he File menu

d. Click OK to create the program file.

3. Double-click the Echo icon in the project tree, and then type the following code in the Editor window:

```
program
    TrunkWaitCall();
    TrunkAnswerCall();
    MediaRecordFile("msg.vox", 15);
    MediaPlayFile("msg.vox");
    TrunkDisconnect();
    restart;
endprogram
```

Compiling and Running the Second Program

Complete the following steps:

1. From the main menu in Graphical VOS, choose Run | Run Echo.vs.

   The program automatically compiles first and then runs. Again, you can see the stopwatch icon and the VOS Box icon in your system tray.

   The program you wrote executes, and then waits for a call to come in.

2. Click Ring in the SimPhone main window.

   You'll see the Line State icon in SimPhone go off-hook, which means that your application has answered the call.

3. Say something into your sound card's microphone and then click any SimPhone digit button.

   This stops the recording, and the message is played back to you. As soon as the message playback has finished, the program ends the call and starts at the beginning. You can then call in again if you want.

4. When you are finished, stop the program by choosing Run | Stop Running from the Graphical VOS main menu.

This completes the tutorial. In the next section you analyze the code in the Echo Program.

Examining the Second Program

In this section we'll dissect the Echo program you created. The following table offers a brief explanation of what happens at each line. The next section provides a more in-depth explanation.

Command	Action
`program`	Start here
`  TrunkWaitCall();`	Wait for call
`  TrunkAnswerCall();`	Answer call
`  MediaRecordFile("msg.vox", 15);`	Record sound to file
`  MediaPlayFile("msg.vox");`	Play back sound
`  TrunkDisconnect();`	Terminate call
`  restart;`	Go back to first statement
`endprogram`	End of main program

Echo Program Code Detail

Every program must begin with the word *program*, and end with the word *endprogram*.

TrunkWaitCall Command

Command: `TrunkWaitCall();`

Action: Waits for an incoming call on the current trunk (phone line)

Description: This command tells VOS to suspend the program until an incoming call is received on the given line. When the call comes in, VOS "wakes up" the program, which then continues execution.

TrunkAnswerCall Command

Command: `TrunkAnswerCall();`

Action: Answers the call on the current trunk line

Description: This command instructs the Dialogic board to “go off hook,” meaning to answer the phone. This action corresponds to picking up the handset on a telephone.

MediaRecordFile Command

Command: `MediaRecordFile("msg.vox", 15);`

Action: Records from the current line to a hard disk file named msg.vox

Description: This command turns the Dialogic board into a digital tape recorder. The sound received on the phone line is digitized and saved to a file on the computer’s hard disk. Just as with MediaPlayFile, there is one mandatory parameter to this command, the hard disk file name to store the digitized sound. In addition to the file name, by setting the second parameter, you can control the maximum length of the recording, in seconds. The example shows MaxSecs set to 15, so the recording can be up to 15 seconds long.

MediaPlayFile command

Command: `MediaPlayFile("msg.vox");`

Action: Plays back sound from msg.vox on the current line

Description: This command plays back sound from a hard disk file. It has one mandatory parameter, which is the name of the file.

TrunkDisconnect command

Command: `TrunkDisconnect();`

Action: Terminates call on the current line

Description: This command finishes the call by putting the line back “on-hook” to hang up.

Restart Command

Command: `restart;`

Action: Goes back to the beginning of the program

Description: This command goes back to the beginning and starts executing again. You can think of it as jumping back to the word *program* and resuming execution from the first command.

Using Basic Telephony Functions

The next two tables list some of the main capabilities of the telephony boards. The following section describes how to write several functions you’re most likely to use.

Audio Processing

This table lists audio processing commands:

Action	Command
Play speech from file	`MediaPlayFile`
Record speech to file	`MediaRecordFile`
Get touch tones from caller	`MediaGetDigitBuffer`

Telephone Signaling

This table lists telephone signaling commands:

Action	Command
Wait for incoming call	`TrunkWaitCall`
Answer incoming call	`TrunkAnswerCall`
Terminate a call	`TrunkDisconnect`
Make outbound call	`TrunkMakeCall`
Detect caller hangup	`onhangup`

Programming Basic Features

This section shows you the code necessary to program basic computer telephony features using VOS script.

Answering an Incoming Call

VOS answers a typical call using the following command sequence:

Action	Command
1. Wait for ring	`TrunkWaitCall();`
2. Go off-hook	`TrunkAnswerCall();`
3. Play greeting	`MediaPlayFile(filename);`

For more information and an example program that answers inbound calls, see the Graphical VOS online help.

Making an Outgoing Call

VOS usually makes an outgoing call using this command sequence:

Action	Command
1. Make outbound call	`TrunkMakeCall();`
2. Check new state	`State = TrunkGetState();`
	`Glare = TrunkGlare();`

The new state of the trunk should be Connected, NoConnect, or InboundRinging. Connected means that the call went through or that a condition called *glare* occurred, that is, an inbound call arrived just as you were attempting an outbound call. You can test for glare with the TrunkGlare function. A NoConnect state means that the call was not completed (for example, there was a busy tone, or the line rang off the hook without being picked up). InboundRinging also indicates a glare condition.

For more information and an example program that makes outbound calls, see the Graphical VOS online help.

Terminating a Call

To end a call, simply use the TrunkDisconnect function. Most programs are designed to start again at the beginning and then prepare to receive another call. You can do this using the restart command:

Action	Command
1. Hang up	`TrunkDisconnect();`
2. Return to start of program	`restart;`

Detecting a Disconnect

The caller may hang up at any point in the VOS program. Consequently, the caller might hang up when the program is not currently waiting for a hang-up or when it's executing an action.

The most commonly used mechanism in VOS is the onhangup function. When a disconnect is detected, VOS interrupts the program wherever it is currently executing, and jumps to the onhangup function. The onhangup function can contain any VOS commands. The onhangup function usually finishes with either a return command or a restart command. A return command sends the program back to the point where it was originally interrupted, and continues exactly as

before (except that variables may be changed in the onhangup function). If a restart command is executed, the program goes back to the start of the program and is ready to receive the next call.

Example 7-1 VOS code that creates an onhangup function

```
onhangup

   TrunkDisconnect();

   restart;

endonhangup
```

Suitable for most VOS applications, this code reacts to a caller hang-up by disconnecting and going back to the beginning of the program, which presumably either waits for the next call to arrive or initiates a new call.

A problem with using onhangup can arise if the program is in the middle of doing something important when the disconnect signal arrives. For example, it may be making updates to a database. This situation can be handled using the TrunkDeferOnHangup and TrunkClearDeferOnHangup commands, which allow a disconnect signal to be *deferred*. Calling TrunkDeferOnHangup says, “I am beginning to do something important—don’t jump to onhangup even if a disconnect signal does arrive.” Calling TrunkClearDeferOnHangup says, “OK, if a disconnect signal did arrive, now jump to onhangup, otherwise just carry on as usual.”

We can add disconnect processing to our example program by adding an onhangup function.

Example 7-2 VOS code that adds an onhangup function

```
program

      voslog("Program started");

      TrunkWaitCall();

      TrunkAnswerCall();

      sleep(10);

      MediaRecordFile("msg.vox", 15);

      MediaPlayFile("msg.vox");

      TrunkDisconnect();

      restart;

endprogram

   onhangup
```

```
      TrunkDisconnect();

      restart;

   endonhangup
```

Creating a Touch Tone Menu

One of the basic tools utilized in automatic call processing is the ability to play a menu and get a response from the caller. A typical audible menu pattern reads like this:

"*Please press 1 to do this, press 2 to do that, press 3 to do...*"

In order for the menu to work correctly, the pre-recorded file containing the voice saying the menu must play first. Then VOS must wait for the caller to enter a touch-tone digit.

Example 7-3 VOS code that creates a touch tone menu

```
 MediaPlayFile("menu.vox");

 MediaWaitDigits(1);

digit = MediaGetDigitBuffer();

switch (digit)

case 1:

   do_one();

case 2:

   do_two();

case 3:

   do_three();

default:

   bad_digit();

endswitch
```

The MediaWaitDigits function waits for the caller to enter one or more digits. The function has one mandatory parameter, that is, the number of digits to get. Thus, MediaWaitDigits(1) waits for a single digit and moves it to the digit buffer (the place where VOS stores digits dialed by the caller).

The MediaGetDigitBuffer command returns the contents of the digit buffer.

In this example we used the commands: do_one, do_two, do_three, and bad_digit. These are *user-defined functions*, that is, new commands you can invent by writing VOS script. User-defined functions are called functions or subroutines in other languages.