Debugging in Graphical VOS 8

This chapter provides a tutorial in which you create and debug an application. It contains the following sections:

- Creating an Application
- Debugging the Application

Note: Debugger is for use with Graphical VOS only.

Creating an Application

In this section, you use Graphical VOS to create an application. In the next section you debug your new application using CT ADE Debugger and SimPhone.

Before You Start

Before beginning this tutorial, make sure CT ADE with Graphical VOS 8.0 or higher is installed correctly. You can run this tutorial in either evaluation or development mode. You don't need a telephony card for this tutorial because you're going to use SimPhone.

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, which includes a description of each VOS function.

Creating a new Project

Complete the following steps:

1. Choose one of the following:
    a. If you don't have Graphical VOS running, launch it from the Start menu and select the Create a new project button on the Welcome to Graphical VOS dialog.
    b. If Graphical VOS is running, close any open projects and choose New from the File menu.
2. In the New dialog:
    a. Select Project for the File Type.
    b. Name the project **Debugger**.

c. Save it in the following directory:

```
C:\Program Files\Parity Software\Graphical
VOS\Tutorial\Debugger directory
```

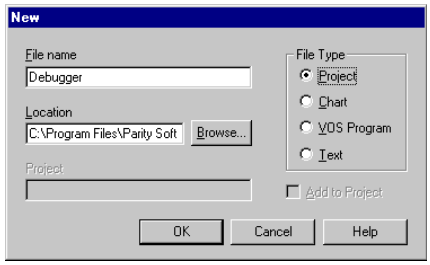


Figure 8-1 New dialog displays when you choose New from the Graphical VOS File menu

d. Click OK to create the project.

Now add a new VOS Program to the project:

3. From the File menu, choose New and then complete the following:

   a. In the New dialog under File Type, select VOS Program.

   b. Name the file **inbound**.

   c. Make sure the Add to Project box is checked.

   d. Click OK to create the file to add it to your Graphical VOS project.

In the Project window, you can see a tree that shows the project's files: a master application, an RLL, and the inbound.vs application:

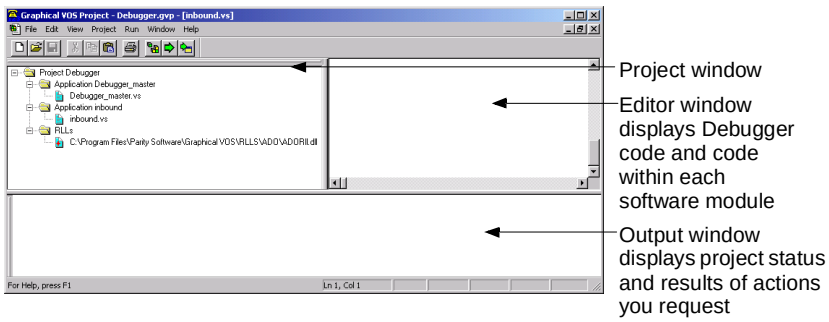


Figure 8-2 Graphical VOS Project window

4. Double-click the inbound.vs icon in this tree, and you can see the empty file in the Editor window to the right.

   The window on the bottom is the output window.

5. Click in the Editor window on the right and type the following code into it:

```
program
  TrunkWaitCall();
  TrunkAnswerCall();
  if (TrunkGetState() strneq "Connected")
     voslog("Unexpected trunk state ", TrunkGetState());
     stop;
  endif
  InboundCall();
  TrunkDisconnect();
  restart;
endprogram


onhangup
  if (TrunkGetState() streq "RemoteDisconnected")
     TrunkDisconnect();
  endif
  restart;
endonhangup

func InboundCall()
  MediaPlayFile("..\Prompts\HelloInbound.vox");
endfunc
```

6. Choose File | Save.

Debugging the Application

Now you're ready to debug the application you created.

Starting Debugger

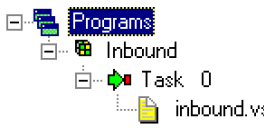
Complete the following steps:

1. In the project window, right-click the Application inbound folder and choose Debug Application.

   Graphical VOS starts Debugger and VOS. The Debugger displays in the right window—which is now called the Tasks window, and VOS displays the VOS Box dialog.

2. Minimize the VOS Box dialog.

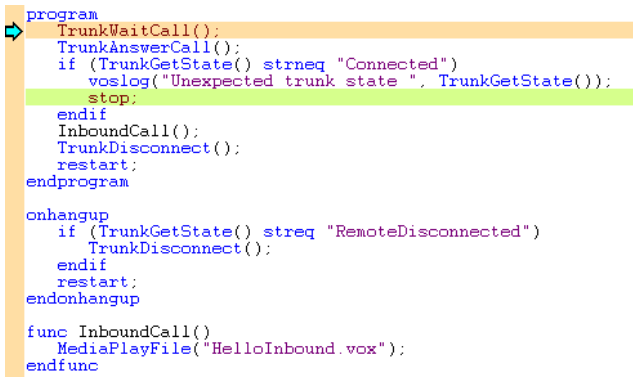
3. If you need to, right-click on the Programs icon in the Tasks window and choose Expand Tasks Tree so that you can see Task 0.



Stepping through Code

Now we can look at the Inbound program more closely.

1. In the Tasks Window, double-click Task 0 to see the source files for the task. Then double-click inbound.vs to switch to the Source File window and see the source code.



```
program
   TrunkWaitCall();
   TrunkAnswerCall();
   if (TrunkGetState() strneq "Connected")
      voslog("Unexpected trunk state ", TrunkGetState());
      stop;
   endif
   InboundCall();
   TrunkDisconnect();
   restart;
endprogram

onhangup
   if (TrunkGetState() streq "RemoteDisconnected")
      TrunkDisconnect();
   endif
   restart;
endonhangup

func InboundCall()
   MediaPlayFile("HelloInbound.vox");
endfunc
```

Figure 8-3 Source code for the inbound.vs task

The next statement to be executed in the task is highlighted and displayed with a “blue arrow” icon. Because we haven’t executed any lines of code yet, the next statement to execute is the first statement in the program:

```
TrunkWaitCall();
```

This waits for an incoming call. The program waits indefinitely until a call arrives.

2. Click the Step Next icon, and the highlight and blue arrow disappear. If you do not see this icon in the toolbar, choose View | Debugger Toolbar.

   VOS is executing the first line. In order for the function to return, the application must receive an incoming call.

3. On the SimPhone main window, click Ring to simulate a call to the voice card.

   The highlight and blue arrow move to the second line in the program:

```
TrunkAnswerCall();
```

4. Click Step Next again and VOS answers the call.

   The blue arrow and highlight move to the next line in the program.

Next we are going to execute a few lines of code without having to click Step Next repeatedly.

Using Breakpoints

A breakpoint is a condition that causes a VOS task to stop execution and enter a Paused state. First let's add a breakpoint to a line:

1. Click once on the line of code that reads:

   ```
   InboundCall();
   ```

   A different highlight, called the cursor, appears. By default, the cursor is bright green.

2. In the toolbar click the Toggle Breakpoint icon and a red hand appears next to the line.

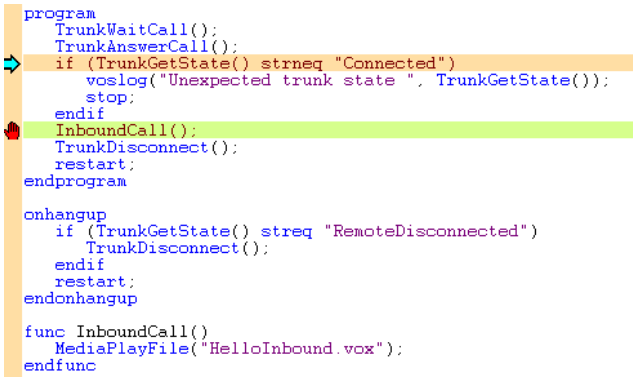


Figure 8-4 The red hand specifies the point at which the Debugger stopped

3. Next, click the Resume icon in the toolbar and the program executes up to the breakpoint.

Viewing the Call Stack

The next line to execute in the main program invokes a user-defined function:

```
InboundCall();
```

To step through this function:

1. Click the Step Into icon.

   Clicking this icon executes up to the first line of the function.

2. Next, click the Call Stack icon to display the Call Stack dialog.

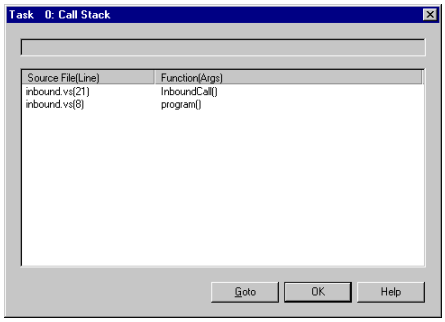


Figure 8-5 Call Stack Dialog

The function call stack is a display showing the hierarchy of function calls that the program went through to get to its current position. You can see that the task is currently paused on line 21 of inbound.vs, in a function called InboundCall. That function was called from line 8 in the main program in inbound.vs.

3. Click OK to close the Call Stack window.
4. Click the Step Next icon to execute the next function, playing the HelloInbound.vox file.

   Finally, close the Debugger.
5. From the Run menu, click Stop Debugging.

For more information about the Debugger, see the Graphical VOS online help.