LabWindows™/CVI™ Core 2 Exercises

Course Software Version 2010
January 2011 Edition
Part Number 325671A-01

Worldwide Technical Support and Product Information
ni.com

National Instruments Corporate Headquarters
11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 662 457990-0, Belgium 32 (0) 2 757 0020, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 5050 9800, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 358 (0) 9 725 72511, France 01 57 66 24 24, Germany 49 89 7413130, India 91 80 41190000, Israel 972 3 6393737, Italy 39 02 41309277, Japan 0120-527196, Korea 82 02 3451 3400, Lebanon 961 (0) 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 (0) 348 433 466, New Zealand 0800 553 322, Norway 47 (0) 66 90 76 60, Poland 48 22 328 90 10, Portugal 351 210 311 210, Russia 7 495 783 6851, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 (0) 8 587 895 00, Switzerland 41 56 2005151, Taiwan 886 02 2377 2222, Thailand 662 278 6777, Turkey 90 212 279 3031, United Kingdom 44 (0) 1635 523545

For further support information, refer to the *Additional Information and Resources* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.

Contents

Lesson 6
LabWindows/CVI Toolkits and Modules

Appendix A
Additional Information and Resources

4

Creating and Using Dynamic Link Libraries (DLLs)

Exercise 4-1 Creating a DLL

Goal

Create a DLL in LabWindows/CVI.

Scenario

In this exercise, you will build a DLL that can be accessed by other programming languages.

Implementation

1. Create a new project.
2. Select **Build»Target Type»Dynamic Link Library**.
3. Create a new source file. Enter the following code or copy and paste the code from `firstdll.c` in the `<Solutions>\CVI Core 2\Exercise 4-1` directory.

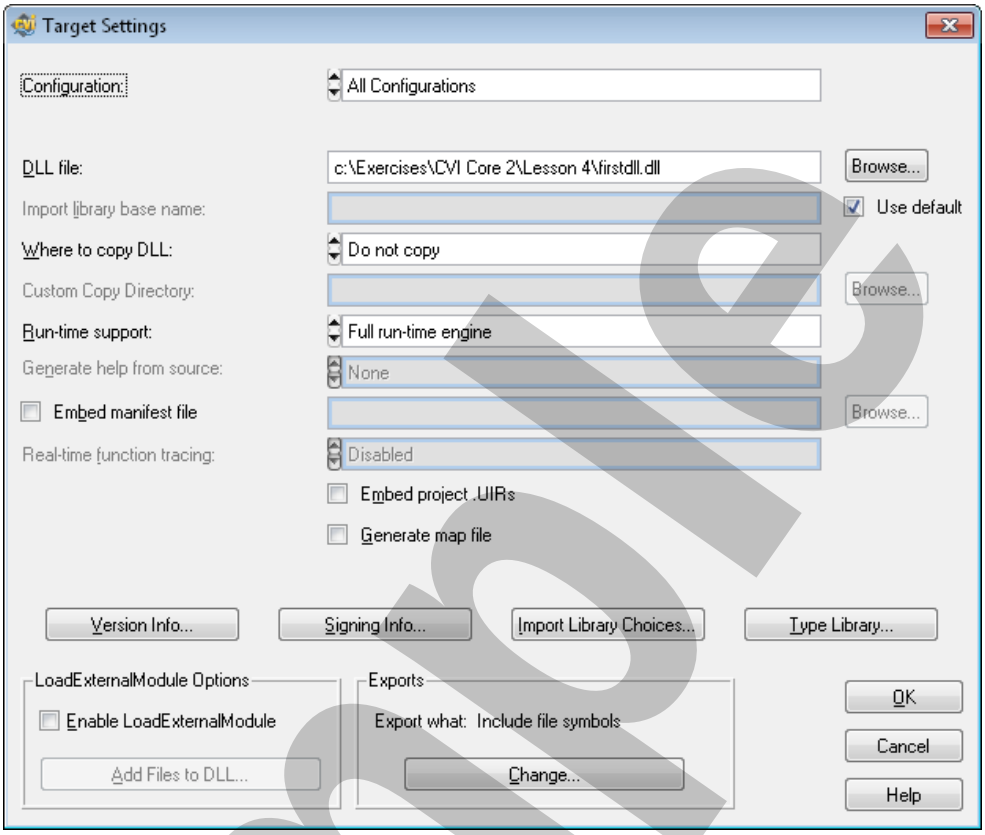
```
#include <utility.h>
#include <cvirte.h>
#include <userint.h>

int _stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason,
   LPVOID lpvReserved)
{
   if (fdwReason == DLL_PROCESS_ATTACH)
   {
      Beep ();
      Delay (.5);
   }
   else if (fdwReason == DLL_PROCESS_DETACH)
   {
      Beep ();
      Delay (.5);
      Beep ();
   }
   return 1;
}
```

```
void fnInternal(void)
{
   MessagePopup ("", "In internal DLL function");
}
void fnDLLTest(void)
{
   MessagePopup ("In DLL Test Function", "Hi there");
   fnInternal();
}
```

Notice that `DllMain` has the `_stdcall` keyword before it. This keyword defines the calling convention for the function. Microsoft Windows contains two calling conventions—the C calling convention, denoted by `__cdecl`, and the standard calling convention, formerly the Pascal calling convention, denoted by `__stdcall`.

4. Save the file as `firstdll.c` in the `<Exercises>\CVI Core 2\Lesson 4` directory and add it to the project.
   - ❑ Create the directory if it does not already exist.
   - ❑ Add `firstdll.c` to the project by selecting **File»Add firstdll.c to Project**.
5. Create a new include (`.h`) file and add it to the project.
   - ❑ Enter the following code in the header file:
     ```
     void fnDLLTest(void);
     ```
   - ❑ Save the file as `firstdll.h` and add it to the project.
6. Save the project as `firstdll.prj`.
7. Select **Build»Configuration»Release**.
8. Select **Build»Target Settings** to open the Target Settings dialog box.



9. Change the target settings to export include file symbols for all configurations, so that any type of build you create will contain the correct export.

   ❑ In the **Exports** section, click **Change** to open the DLL Export Options dialog box.

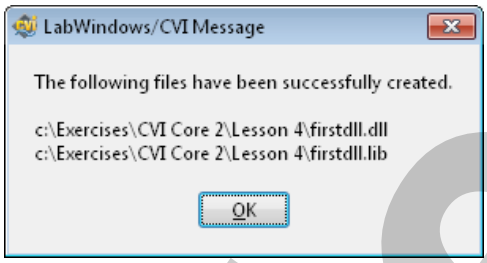
   ❑ In the DLL Export Options dialog box, select `firstdll.h`, and click **OK**. LabWindows/CVI uses this header file to determine what functions to export. The only function prototype in the header file is for `fnDLLTest`, so it is the only function exported. Refer to the *Target Settings for DLLs* topic of the *LabWindows/CVI Help* for more information about export settings.

   ❑ Set **Configuration** to **All Configurations**.

   ❑ Click **OK** to close the Target Settings dialog box and return to the Workspace window.

10. Select **Build»Create Release Dynamic Link Library**. Click **OK** if prompted to set debugging to none.

11. After the build completes, the following message displays.



By default, the DLL and import libraries created have the same base name as the project. In this case, `firstdll.dll` is created along with `firstdll.lib`, the import library.

12. Click **OK** in the LabWindows/CVI message, save any changes, and close all open windows except the Workspace window.

End of Exercise 4-1

Exercise 4-2 Using a DLL

Goal

Use the DLL created in the previous exercise.

Implementation

In this exercise, you work with two projects in the same Workspace.

1. Create a new project in the current workspace.
2. Save the project as `testdll.prj` in the `<Exercises>\CVI Core 2\Lesson 4\` directory.
3. Create a new source file. Enter the following code in the Source window:

```
#include "firstdll.h"
main()
{
   fnDLLTest();
}
```

4. Save the source file as `testdll.c` and add the file to the project.
5. Add the import library to the project so that the function can be resolved.
   - ❑ Select **Edit»Add Files to Project»Library (*.lib)**.
   - ❑ In the Add Files to Project dialog box, select `firstdll.lib`.
   - ❑ Click **Add**, then click **OK**.
6. Run the project.
   - ❑ Two message pop-ups appear.
     - One for the call to `fnDLLTest`
     - One for the internal call from `fnDLLTest` to `fnInternal`
   - ❑ Three beeps sound.
     - One for the load (`DLL_PROCESS_ATTACH`) of the DLL
     - Two for the unload (`DLL_PROCESS_DETACH`) of the DLL

Note If you are using a machine without a sound card, the beeps will not be audible.

7. In `testdll.c`, add the following bolded lines to the code:

```
#include "firstdll.h"
void fnInternal(void);
main()
{
   fnInternal();
   fnDLLTest();
}
```

8. Run the project again. Recall that `fnInternal` was not exported when you created the DLL. Therefore, when you try to run the project, you get an `"Undefined symbol"` error when the project is linked.

9. Delete the code you added in step 7.

10. Save and close `testdll.c`.

End of Exercise 4-2

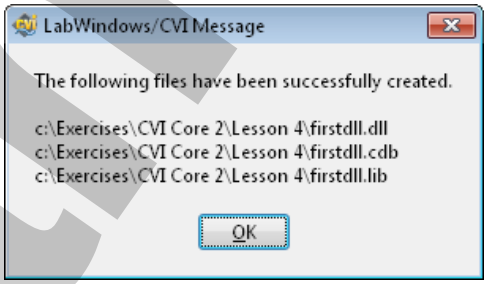
Exercise 4-3 Examining DLL Debugging Features

Goal

Use the DLL debugging features.

Implementation

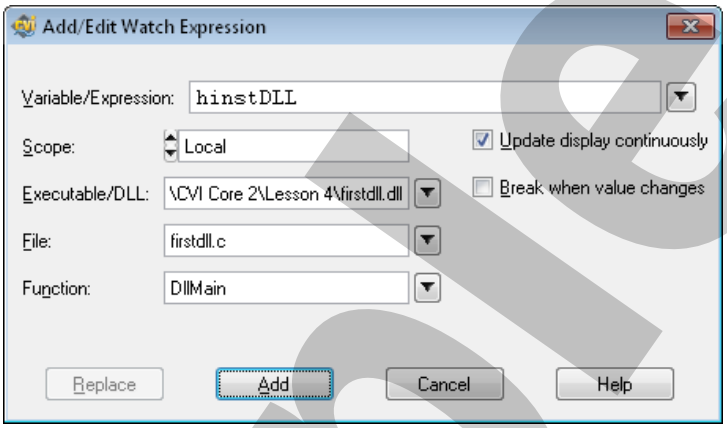
1. Set `firstdll.prj` as the active project.
2. Select **Build»Configuration»Debug** to enable DLL debugging.
3. Select **Build»Target Settings** to open the Target Settings dialog box and verify the export setting.
   - ❑ In the **Exports** section, click **Change** to open the DLL Export Options dialog box.
   - ❑ Verify that `firstdll.h` is selected, and click **OK**.
   - ❑ Click **OK** to close the Target Settings dialog box.
4. Rebuild the DLL. Select **Build»Create Debuggable Dynamic Link Library**, overwriting the existing DLL and import library files.
5. After the build completes, the following window displays.



6. Click **OK** in the LabWindows/CVI message.
7. Set `testdll.prj` as the active project.
8. Open `firstdll.c`, the source file for the DLL created in step 4, in the same workspace.
9. Place your cursor on the first detach beep, around line 15 and select **Run»Toggle Breakpoint** to insert a breakpoint.

10. Run the `testdll.prj` project. LabWindows/CVI breaks at the appropriate statement in the DLL source file.

    - ❑ You can add watch expressions for variables in the DLL through the Watch window, as shown in the following figure.



    - ❑ Try adding watch expressions for other variables and see the different settings applicable for variables defined in DLLs.

11. After testing the different DLL debugging features, close all files.

End of Exercise 4-3

Notes

Notes