

# LabVIEW™

## Embedded Development Module Target Distribution Guide

## **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 385 (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 413091, Japan 81 3 5472 2970, 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 3390150, 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 *Technical Support and Professional Services* appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at [ni.com/info](http://ni.com/info) and enter the info code `feedback`.

# Important Information

---

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

## Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on [ni.com/legal](http://ni.com/legal) for more information about National Instruments trademarks.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

## Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or [ni.com/patents](http://ni.com/patents).

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Conventions

---

The following conventions are used in this manual:

[ ] Square brackets enclose optional items—for example, [response].

» The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.



This icon denotes a tip, which alerts you to advisory information.



This icon denotes a note, which alerts you to important information.

**bold** Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

*italic* Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text also denotes text that is a placeholder for a word or value that you must supply.

`monospace` Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

*monospace italic* Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

# Contents

---

## Chapter 1

### Distributing Your Embedded Target and LabVIEW

Distribution Options .....	1-1
Process Overview of Bundling Your Embedded Target with LabVIEW .....	1-2

## Chapter 2

### Wrapping Drivers and Math Libraries

Creating SubVIs for Target-Specific Functions .....	2-1
Using a Call Library Function Node in a SubVI .....	2-2
Using an Inline C Node in a SubVI .....	2-3
Using the Import Shared Library Wizard .....	2-3
Modifying Funclist.dat to Replace a SubVI .....	2-3
Naming VIs .....	2-4

## Chapter 3

### Creating Icons and Palettes

Creating Icons .....	3-1
Customizing a Palette Set for Your Target .....	3-2
Creating Subpalettes in a Palette Set .....	3-3
Adding Objects to Subpalettes .....	3-4
Correcting Missing Objects on Palettes .....	3-4

## Chapter 4

### Creating Examples for Your Target

Example Style Guidelines .....	4-1
General Example Construction .....	4-1
Front Panel Design .....	4-2
Block Diagram Design .....	4-2
SubVI Construction .....	4-3
Adding Examples to the NI Example Finder .....	4-3

## Chapter 5

### Adding Target-Specific Documentation

Documentation Directory Structure .....	5-2
Documentation Naming Conventions .....	5-2
Help Filenaming Tips.....	5-2
PDF Filenaming Tips .....	5-2
Readme Filenaming Tips .....	5-3
Reference and Non-Reference Help.....	5-3
Creating VI Reference .....	5-3
Creating Object Descriptions and Tip Strips for VIs and Dialog Boxes .....	5-5
Linking VIs to HTML Files or Compiled Help Files .....	5-6
Linking Dialog Box Help Buttons to	
Compiled Help Files or HTML Files .....	5-7
Linking to a Topic in a Compiled Help File.....	5-7
Linking to an Independent HTML File .....	5-7
Documenting Concepts and Tasks .....	5-8
Putting Your CHM Help File in the LabVIEW Help Menu .....	5-8
Printed and PDF Manuals.....	5-8
Readme Files .....	5-8

## Chapter 6

### Testing Your Target

Manually Testing Your Target.....	6-1
Using the Automated Test Framework.....	6-1

## Chapter 7

### Licensing Your Target and Creating the Installer

Signing Your Target.....	7-1
Listing the Plug-In VIs for Signing.....	7-1
Sending Files to National Instruments for Signing and Licensing .....	7-2
What National Instruments Does with Your Target .....	7-2
Creating an Installer for Your Target .....	7-3
Installing the Necessary Components for Your Target.....	7-3
Locating the LabVIEW 8.5 Directory.....	7-3
Options for End Users Installing Your Product .....	7-4
Running LabVIEW in Evaluation Mode .....	7-4
Running LabVIEW in Licensed Mode .....	7-4

## Appendix A

### Technical Support and Professional Services

---

# Distributing Your Embedded Target and LabVIEW

The LabVIEW Embedded Development Module enables OEMs/VARs and advanced users to port LabVIEW to embedded devices from any manufacturer. LabVIEW refers to these embedded devices as targets. Typical targets contain 32-bit microcontrollers with an embedded operating system and 256 KB RAM or more. After you create a new target and port LabVIEW to support that target, you can use LabVIEW to develop embedded applications and make LabVIEW and your target available for other end users.

This document describes other things you might want to do after you port LabVIEW to a new target to increase the usability of your target in LabVIEW. This document assumes you are familiar with LabVIEW and the Embedded Development Module.

For information about porting LabVIEW to your target, refer to the *LabVIEW Embedded Development Module Porting Guide*, available by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals** and opening `EMB_Porting_Guide.pdf`.

---

## Distribution Options

The process of distributing your target and LabVIEW can be minor or extensive, depending on your business need. You might zip up the VIs you create or modify for your target, write some documentation about how to use your target with LabVIEW, and email the zip file and documentation to a few associates. The associates must have a licensed version of the LabVIEW Embedded Development Module to use the VIs with your target.

The other end of the distribution spectrum includes obtaining an agreement from National Instruments to bundle and resell LabVIEW and your target. This document guides you through the process of creating a product for your target that you can bundle with LabVIEW and resell to end users. You also can use some of the tips and suggestions in this document to enhance

the end user experience with your target, even if you do not plan to formally package your product.

## Process Overview of Bundling Your Embedded Target with LabVIEW

---

The following list outlines the process of formally creating a product for your new embedded target and bundling that product with LabVIEW.



**Note** National Instruments requires that you have these items in place before you resell LabVIEW and your target.

1. The OEM/VAR creates a custom target and ports LabVIEW to the new target based on the information in the *LabVIEW Embedded Development Module Porting Guide*.
2. The OEM/VAR enhances the end-user experience in the following ways.
  - Wrapping drivers, VIs, and math libraries
  - Creating custom LabVIEW icons and palettes
  - Creating examples applications
  - Adding target-specific documentation
3. The OEM/VAR thoroughly tests and debugs the target.
4. After the target is ready for distribution, the OEM/VAR sends the `TgtSupp.xml` file they create to National Instruments for signing so end users cannot modify the target VIs.
5. National Instruments sends the OEM/VAR the signed `.xml` file, product DLL, and custom license file so the OEM/VAR can create an installer for the product.
6. The OEM/VAR orders special LabVIEW CDs from National Instruments.
7. The OEM/VAR burns installer CDs for the product and bundles these product CDs, LabVIEW CDs, and other items for the final package.



**Tip** Your National Instruments contact can help you establish time estimates for accomplishing each of these tasks.

---

# Wrapping Drivers and Math Libraries

Applications for an embedded target often use C-based libraries and functions that are unique to that particular embedded target. For example, a target might have specific libraries for accessing I/O, optimized math functions, or other unique features such as a bar code reader or LCD.

One way end users can access these target-specific functions in LabVIEW is by using a Call Library Function Node in their application every time they want to call one of these functions directly. However, this process can add unnecessary complexity to an application.

You can make the task of calling these functions easier for the end user by wrapping C-based target-specific functions into subVIs, which are VIs called from the block diagram of other VIs.

Wrapping functions into subVIs is a good option if you plan to bundle your embedded target with LabVIEW and resell the product. Being able to use subVIs instead of C-based target-specific functions is an attractive product feature, especially for end users without a lot of embedded development experience.

---

## Creating SubVIs for Target-Specific Functions

---

LabVIEW provides the following options for wrapping C-based functions into subVIs:

- Use a Call Library Function Node in a subVI.
- Use an Inline C Node in a subVI.
- Use the Import Shared Library Wizard.
- Modify the `Funclist.dat` file to substitute a C function call for a subVI.

If you use the Call Library Function Node or Inline C Node, you must manually build a connector pane and create an icon for the VI to create the subVI. Refer to Chapter 3, [Creating Icons and Palettes](#), for information

about icon creation. You also can refer to the *Creating SubVIs* topic in the *LabVIEW Help* for information about creating connector panels and icons for subVIs.



**Tip** The Import Shared Library Wizard builds a connector pane and creates an icon for the VI by default.



**Note** Because embedded target-specific functions cannot run on a Windows target, you might want to use a Conditional Disable structure in wrapper VIs to add an alternative implementation for Windows targets. Adding an implementation for Windows targets gives the end user more options for debugging an application and allows the end user to test parts of an application without connecting the target device to the computer. Refer to the *LabVIEW Help* for information about using Conditional Disable structures.

## Using a Call Library Function Node in a SubVI

Rather than requiring end users to use a Call Library Function Node every time they want to call C-based functions, DLLs, or shared libraries, you can create a subVI that uses a Call Library Function Node to make the call. End users then use your subVI in the same way they use other subVIs in LabVIEW, which is helpful for LabVIEW programmers who might not also be text-based programmers.

To create a subVI that calls a C-based function, place a Call Library Function Node on a blank block diagram and use a function prototype that matches the C function you want to call. Call Library Function Nodes consist of pairs of input and output terminals. Right-click the node and select **Configure** from the shortcut menu to display the **Call Library Function** dialog box, which you can use to specify the library name or path, function name, calling conventions, parameters, and return value. When you click the **OK** button, the node automatically resizes to have the correct number of terminals and sets the terminals to the correct data types.

After you configure the Call Library Function Node, right-click each input terminal and select **Create»Control** from the shortcut menu. Right-click each output terminal and select **Create»Indicator** from the shortcut menu.

Refer to the *Using External Code in LabVIEW* topic in the *LabVIEW Help* for information about Call Library Function Nodes.

## Using an Inline C Node in a SubVI

If the code to call is not a C-based function—for example, assembly code or a small algorithm—consider using an Inline C Node instead of a Call Library Function Node. You can use an Inline C Node for any C code, including assembly directives and `#defines`, that syntactically is between the curly braces. The concept is the same as using a Call Library Function Node in a subVI so end users can use your subVI in the same way they use other subVIs in LabVIEW.

Right-click the left side of the Inline C Node and select **Add Input** from the shortcut menu for each input you want to add to the Inline C Node. Right-click the right side of the Inline C Node and select **Add Output** from the shortcut menu for each output you want to add. For each input you add, right-click the terminal and select **Create»Control** from the shortcut menu. For each output you add, right-click the terminal and select **Create»Indicator** from the shortcut menu.

Refer to the *Inline C Node* topic in the *LabVIEW Help* for information about the Inline C Node.

## Using the Import Shared Library Wizard

You can use the Import Shared Library Wizard to create wrapper VIs based on C functions. Refer to the *Importing Functions from a Shared Library File* topic in the *LabVIEW Help* for information about using the Import Shared Library Wizard. You must place a checkmark in the **Shared library file is not on the local machine** checkbox to create wrapper VIs for embedded targets.



**Note** The Import Shared Library Wizard builds a connector pane and creates an icon for the VI by default, so the VI is ready immediately to use as a subVI.

## Modifying FuncList.dat to Replace a SubVI

The `FuncList.dat` file, located in the `labview\CCodeGen` directory, contains lines of code fragments that LabVIEW substitutes for a subVI call when the LabVIEW C Code Generator builds C code for the application. The code fragments can be anything but are usually function calls. Adding entries to `FuncList.dat` makes the application run faster because using `FuncList.dat` removes the overhead of calling a subVI that then calls the functions using a Call Library Function Node or Inline C Node.

You still need to create a subVI for the block diagram before you can use `FuncList.dat`, but the VI can be empty except for the necessary controls and indicators corresponding to the parameters for the function.



**Tip** Consider first creating a subVI with a Call Library Function Node or an Inline C Node to make debugging the input and outputs of the function easier. After you get the subVI working correctly in an application, you then can modify the `FuncList.dat` file with the substitute code to improve the performance of the application.

Refer to the *LabVIEW Embedded Development Module Porting Guide*, available by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals** and opening `EMB_Porting_Guide.pdf`, for more information about `FuncList.dat`.

## Naming VIs

---

VI names do not have the same limitations as C function names, so make VI names descriptive. Descriptive names make it easy to identify and use a VI on a block diagram.

National Instruments also strongly recommends that you create unique filenames for VIs. One way you can make sure a filename is unique is to prepend a short name or abbreviation to the VI name. In addition to keeping VI names unique, adding an identifier to the VI name also helps end users find a VI that is specific to your target. For example, the filenames for Analog Devices Blackfin targets start with `BF` to keep the names unique and easily identifiable.

---

# Creating Icons and Palettes

Creating custom icons and palettes for your product enhances the product identity and makes it easier for end users to identify controls, indicators, and VIs that are specific to your embedded target.

LabVIEW represents VIs and functions with icons, which appear on subpalettes and the block diagram. Icons for LabVIEW modules often have a unique banner or style that visually reminds the end user that the VI or function is specific to that module.

Depending on the target, LabVIEW shows different palettes or palette contents so that end users do not use unsupported VIs or functions in applications for a specific target. You can create palettes to include only the controls, indicators, VIs, and functions that your embedded target supports. With subpalettes you can categorize VIs and functions into logical groups so it is easier for your end user to use LabVIEW with your target.

---

## Creating Icons

An icon is a graphical representation of a VI that appears on the block diagram. The most useful icons are mnemonic and include little, if any, text. Every VI displays an icon in the upper right corner of the front panel and block diagram windows. If you use a VI as a subVI, the icon identifies the subVI on the block diagram of the VI.

Create custom icons to replace the default icon in the upper right corner of the front panel and block diagram windows by selecting **Edit Icon** from the shortcut menu or by double-clicking the icon in the upper right corner of the window. You also can drag a graphic from anywhere in your file system and drop it in the upper right corner of the front panel or block diagram window. LabVIEW converts the graphic to a 32 × 32 pixel icon. Depending on the type of monitor your end users use, you can design a separate icon for monochrome, 16-color, and 256-color mode. You can use the `lv_icon` VI template in the `labview\resource\plugins` directory to create a VI with which you can edit VI icons outside of the **Icon Editor** dialog box. Refer to the Icon Art Glossary at [ni.com](http://ni.com) for standard graphics you can use in a VI icon.

Consider creating icons for your target-specific VIs using consistent colors or themes. If you plan on reselling integrated target support for LabVIEW, you might consider outsourcing the icon creation if you do not have access to professional graphic artists.

Refer to the *Creating an Icon* topic in the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for more information about creating VI icons.

## Customizing a Palette Set for Your Target

---

National Instruments strongly recommends that you remove VIs and functions from the default palette set for your target that your target cannot support. Custom palette sets help end users quickly find the VIs and functions they need for an application and help end users avoid using VIs and functions that your target cannot support.



**Tip** You might want to create your target-specific palette based on the palette set for the Embedded Development Module because the Embedded Development Module palette set already removes VIs and functions that embedded targets do not support. Copy the menu files located in `Targets\NI\Embedded\common\TargetInfo\menus` to the menu directory for your target. You specify the menu directory for your target in the **Target Properties** dialog box in the Target Editor, available by selecting **Tools»Embedded Tools»Target Editor**.

National Instruments also strongly recommends you create subpalettes for the target-specific VIs you create. Otherwise, end users must access your VIs by clicking **Select a VI** on the **Functions** palette and browsing to the directory containing the VI.

Complete the following steps to edit a palette set for a specific target.

1. Create a blank VI or open an existing VI under the target whose palette set you want to modify.
2. Select **Tools»Advanced»Edit Palette Set** to display the **Edit Controls and Functions Palette Set** dialog box.
3. Right-click the **Controls** or **Functions** palette and select from the options on the shortcut menu.

Right-click an open area on the palette to insert an object, row, or column and toggle the standard menu view. Right-click an icon to move a subpalette, delete a subpalette, copy an icon, edit a subpalette icon, rename a subpalette, or edit an icon short name. To move an object, drag it to a new location on the palette.

4. After you edit the palettes, click the **Save Changes** button on the **Edit Controls and Functions Palette Set** dialog box. Click the **Cancel** button to cancel any changes you made in this editing session and close the dialog box.

If you edit the palettes, LabVIEW saves the edits in a copy of the original palette. LabVIEW puts the copy in the `Palettes` folder in the default data directory—for example, `My Documents\LabVIEW Data\8.5\Palettes`. The protection of the original palettes ensures that you can experiment with the palettes without corrupting the original palettes. To revert to the original palettes, click the **Restore to Default** button on the **Edit Controls and Functions Palette Set** dialog box.



**Tip** To edit the original palette set for your target, add the line `InternalPaletteEdit=True` to the `LabVIEW.ini` file located in the `labview` directory before customizing the palette set for your target. Consider creating a backup copy of the palette set before making changes in case you want to revert the changes later.

## Creating Subpalettes in a Palette Set

Complete the following steps to insert a new, empty subpalette on the **Controls** or **Functions** palette. After you create a subpalette, you can add objects to the subpalette.

1. Select **Tools»Advanced»Edit Palette Set** to display the **Edit Controls and Functions Palette Set** dialog box.
2. Right-click the palette and select **Insert»Subpalette** from the shortcut menu to display the **Insert Subpalette** dialog box.
3. Select **Create a new palette file** to insert a new, empty subpalette and click the **OK** button.
4. In the file dialog box that appears, enter the `.mnu` filename. The name of the main subpalette for your target must be 18 characters or less to prevent the main LabVIEW **Functions** palette from automatically widening. Add a `.mnu` extension to the filename to indicate that it is a palette. You must store each subpalette you create in a separate `.mnu` file. Save the `.mnu` files in a folder using your target name in the `labview\menus` directory.
5. Click the **Save** button. The **Edit Palette Item Name** dialog box appears.
6. Enter the name of the new subpalette and click the **OK** button. The new, empty subpalette appears on the palette.



**Note** Empty subpalettes are only visible when you edit a palette set.

## Adding Objects to Subpalettes

After you create a subpalette, you can add objects to the subpalette. A subpalette can contain VIs, controls, and subpalettes.



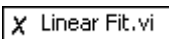
**Note** The **Functions** palette is for VIs and functions that end users use as subVIs. Do not add examples to the **Functions** palette. Refer to Chapter 4, *Creating Examples for Your Target*, for information about including examples with your product.

Complete the following steps to add objects to a subpalette.

1. Select **Tools»Advanced»Edit Palette Set** to display the **Edit Controls and Functions Palette Set** dialog box.
2. Click the subpalette icon to display the subpalette.
3. Right-click the subpalette and select **Insert»Custom Control(s)** or **Insert»VI(s)** from the shortcut menu.
4. In the file dialog box that appears, navigate to the object you want to add to the subpalette.
5. Select the object you want to add to the subpalette and click the **OK** button.
6. Repeat steps 3 through 5 for each object you want to add to the subpalette. You can add objects from different directories on disk to the same subpalette.
7. Click the **Save Changes** button in the **Edit Controls and Functions Palette Set** dialog box to save the changes.

## Correcting Missing Objects on Palettes

During development of your product, you might move VIs without updating the palette set.



If the palette uses a text view, LabVIEW displays objects with an X mark, shown at left, on a pinned palette and dims objects on a temporary palette to indicate that LabVIEW cannot find the object.



If the palette uses an icon view, LabVIEW displays a question mark on the palette, shown at left, to indicate that LabVIEW cannot find the object.

To correct the palette set, complete the steps in the previous section or move the VI back to the original directory on disk.

---

# Creating Examples for Your Target

LabVIEW users often rely on examples to get started with an application because they can start with something that works on the target and gradually modify and test the example to learn and further their understanding.

National Instruments recommends that you create at least one example for each feature and peripheral for your target. Make the examples relatively simple, well documented, and well commented so end users can learn from the examples. Focus an example on a single feature. Create separate additional examples that combine features to illustrate interfaces and communication.

## Example Style Guidelines

---

The following guidelines can help you create consistent, easy-to-use examples:

### General Example Construction

- Name VIs, controls, and terminals with descriptive names. Example VIs that ship with LabVIEW have titles that describe the concept or feature that the example demonstrates. LabVIEW examples do not include the word `example` in the title.
- Do not overuse local and global variables because they can hide data dependency among VIs and might introduce race conditions.
- Mark the appropriate VI as top-level.
- Make sure the VI is not set to run when opened so the end user can explore the front panel and block diagram before running the VI.
- Use an appropriate icon. Do not use the default LabVIEW icon, a blank icon, an icon from a VI in `vi.lib`, or a text-only icon.
- Document the example. In addition to code comments, create a description of the example that explains what the example does. Select

**File»VI Properties** and select **Documentation** from the **Category** pull-down menu. Enter the description in the **Documentation** text box. National Instruments recommends making the example descriptions similar in construction. For example, you might use the following phrase at the beginning of each example description: `This example shows how to....`

## Front Panel Design

- Try to fit the user interface on one screen. Ideally, front panel windows are less than  $800 \times 600$  unless truly necessary. If you must make the front panel window larger, the example still must be usable on an  $800 \times 600$  monitor.
- Open the example in the top-left corner of the screen. Save the example so the **Project Explorer** window, the front panel window, and the block diagram window cascade from the top-left corner with the titlebars visible.
- Use fonts and colors consistently, and use colors sparingly. Use the application font to achieve the best portability.
- Save the VI with appropriate default values in the controls. The VI must run with default values. Also, specify a data range, if possible.
- Use the most recent controls. Do not use the Classic controls on the **Controls** palette.
- Save examples without data in graphs or charts. Right-click the graph or chart and select **Data Operations»Clear Graph** to remove data from the indicator.

## Block Diagram Design

- Avoid unnecessary coercion of data types.
- Avoid placing any wires under block diagram objects such as subVIs or structures.
- Avoid unnecessary bends in the wires and try to keep the wires short. End users might find wires with long, complicated paths confusing.
- Create block diagrams that follow a left-to-right programming style. If the example is larger than the block diagram and you cannot avoid scrolling, scrolling left to right is preferable over scrolling top to bottom.
- Place comments throughout the code. You can double-click anywhere on the block diagram to create a comment string. In the comments, explain what is happening in the application and why.

- Turn on labels for all VIs and functions. Select a VI or function on the block diagram, right-click, and select **Visible Items»Label** from the shortcut menu to turn on labels. National Instruments recommends turning on the labels as you create the example rather than turning on the labels at the end to avoid having to rearrange the block diagram.

## SubVI Construction

- Use the same connector pane pattern on all subVIs.
- Place controls (inputs) on the left and indicators (outputs) on the right in the connector pane to maintain the left-to-right layout and data flow design of most LabVIEW examples.

## Adding Examples to the NI Example Finder

---

You can add the examples you create to the NI Example Finder so your examples are integrated with the rest of the LabVIEW examples. Refer to the *Preparing Example VIs to Appear in the NI Example Finder* topic in the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for information and specific instructions about adding your examples to the NI Example Finder.

---

# Adding Target-Specific Documentation

National Instruments recommends that you provide target-specific documentation with your product so end users know how to use your target with LabVIEW. The following list describes suggested documentation for your embedded target:

- **Reference help**—Context help and compiled help topics describe VIs and dialog boxes.
- **Non-reference help**—Compiled help topics include information about using your target with LabVIEW such as important concepts and instructions for completing common tasks.
- **Printed manuals**—Printed manuals provide, at a minimum, installation instructions and system requirements.
- **PDF manuals**—Soft-copy versions of printed manuals provide back-up copies of printed information. You also can include PDF-only manuals that your end users can print out.
- **Readme**—Readme files contain important last-minute information and known issues.

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for more information about creating documentation. Specifically, refer to the following books on the **Contents** tab:

- **Fundamentals»Documenting and Printing VIs**
- **Fundamentals»Development Guidelines»Concepts»Creating Documentation**

## Documentation Directory Structure

---

LabVIEW documentation uses the following directory structure:

- `labview\help` contains the help files (.chm) for the LabVIEW products you have installed.
- `labview>manuals` contains PDF documents for LabVIEW products you have installed. Some of the PDF documents also are available in print, and some are available in PDF only.
- `labview\readme` contains the readme files for LabVIEW products you have installed. Readme files usually contain known issues and important last-minute and additional documentation.

## Documentation Naming Conventions

---

Your documentation filenames (help, PDFs, and readme files) must have unique filenames. You cannot use the same filename as a standard LabVIEW, LabVIEW module, or LabVIEW toolkit document.

### Help Filenaming Tips

Use the following guidelines when creating help for your target:

- Try to make HTML filenames, including the .html extension, 30 characters or less.
- Use underscores instead of spaces in HTML filenames and the CHM filename.
- Use an identifier to ensure you are not creating HTML filenames that are the same as LabVIEW. For example, most HTML filenames in the Embedded Development Module help start with `emb_`.

### PDF Filenaming Tips

Use the following guidelines when creating PDFs for your target:

- Use a naming convention similar to the one that LabVIEW products use. LabVIEW products use `PRODUCT_Manual_Title.pdf` where `PRODUCT` is a short name identifying your product. For example, the *LabVIEW Embedded Development Module Porting Guide* filename is `EMB_Porting_Guide.pdf`.
- Capitalize the first letter of each word in the manual title.
- Use underscores for spaces.
- Ignore prepositions and conjunctions in the manual title.

- Do not use obscure filenames. Remember that end users can access the PDF files through **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals**, which opens a **File Explorer** window. Descriptive filenames help end users find the manual they are looking for more easily.

## Readme Filenaming Tips

Use the following guidelines when creating readme files for your target:

- Use a naming convention similar to the one that LabVIEW products use. LabVIEW products use `readme_PRODUCT.html` where *PRODUCT* is a short name identifying the product. For example, the readme filename for the Embedded Development Module is `readme_EMB.html`.
- Do not use `readme.html` as the filename. This filename is the LabVIEW development system readme file.

## Reference and Non-Reference Help

---

Help can consist of reference and non-reference documentation.

Reference documentation, which also is called functional documentation, provides descriptive information about objects you add to LabVIEW, such as VI reference and dialog box reference topics.

Non-reference documentation, which also is called conceptual or task-based documentation depending on the content, provides information about your target and how to use it in LabVIEW.

## Creating VI Reference

You can use LabVIEW to document the target-specific VIs you add to LabVIEW. Create descriptions for VIs to describe the purpose of the VI and to provide instructions for using the VI. The VI description appears in the **Context Help** window when you move the cursor over the VI icon in the palette or on the block diagram.

Complete the following steps to create and edit the description of a VI as well as the VI controls and indicators.

1. Select **File»VI Properties** to display the **VI Properties** dialog box.
2. Select **Documentation** from the **Category** pull-down menu.

3. Enter or edit the description of the VI in the **VI Description** text box.

The VI description should include information about the controls and indicators. Consider including the following information in the description:

- Functionality
- Data type
- Valid range (for inputs)
- Default value (for inputs) unless you list the default value in parentheses as part of the control name
- Behavior for special values (0, empty array, empty string, and so on)
- Additional information, such as if the end user must set this value always, often, or rarely



**Tip** Use `<B>` and `</B>` tags to format text as bold. `<B>` and `</B>` are the only HTML tags that work in the **Context Help** window. Lowercase `<b>` and `</b>` do not format text as bold in the **Context Help** window.

4. (Optional) Link from the VI to an HTML file or compiled help file. Refer to the [Linking VIs to HTML Files or Compiled Help Files](#) section for information about linking documentation to a VI.
5. Click the **OK** button to save the changes to the VI properties.



**Tip** You also can use the VI Description property to create and edit the VI description programmatically.

6. Designate required, recommended, and optional VI inputs and outputs to prevent end users from forgetting to wire subVI connections. To indicate the required, recommended, and optional inputs and outputs, right-click the connector pane, select **This Connection Is** from the shortcut menu, and select **Required**, **Recommended**, or **Optional**.

## Creating Object Descriptions and Tip Strips for VIs and Dialog Boxes

Create descriptions for objects, such as controls and indicators, to describe the purpose of the object and to provide instructions for using the object. Create descriptions for controls and indicators in target-specific VIs as well as dialog boxes you create. The object description appears in the **Context Help** window when you move the cursor over the VI icon on the block diagram or in the palette and in VI documentation you generate. Tip strips, also commonly called ToolTips, are brief descriptions that appear when you move the cursor over an object while a user interface VI, such as the **Build Specification** dialog box subpanel, runs.

Complete the following steps to enter a description and a tip strip for an object.

1. Place an object on the front panel or block diagram.
2. Label the object.
3. Right-click the object and select **Description and Tip** from the shortcut menu to display the **Description and Tip** dialog box.
4. Enter a description of the object in the **Description** text box. The **Description** text box automatically includes the label you entered for the object in step 2. Use `<B>` and `</B>` tags to format text as bold in the **Context Help** window.



**Tip** Lowercase `<b>` and `</b>` do not format text as bold in the **Context Help** window. `<B>` and `</B>` are the only HTML tags that work in the **Context Help** window.

5. Enter a tip strip for the object, up to 80 characters, in the **Tip** text box. The **Tip** text box includes the label you entered for the object in step 2.



**Note** The **Tip** text box is available only for controls, indicators, or constants. You cannot enter tip information for VIs or functions.

6. Click the **OK** button.

You also can use the Description and Tip Strip properties to change description and tip information programmatically.

## Linking VIs to HTML Files or Compiled Help Files

If you link a VI to an HTML file or to a compiled help file, a **Detailed Help** link to that file appears in the **Context Help** window when you move the cursor over the VI icon. The end user can get more information about the VI quickly and easily by clicking the **Detailed Help** link.

Complete the following steps to link a VI to an HTML file or to a compiled help file.

1. Select **File>VI Properties** to display the **VI Properties** dialog box.
2. Select **Documentation** from the **Category** pull-down menu as shown in Figure 5-1.

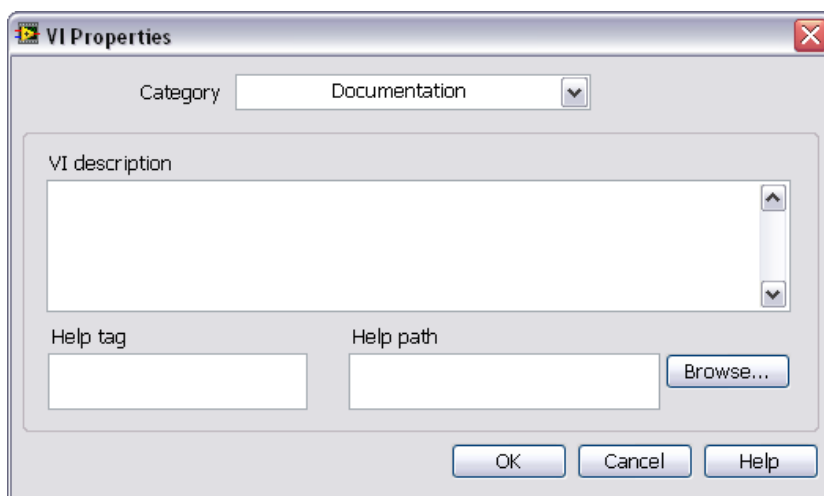


Figure 5-1. VI Properties Dialog Box

3. Click the **Browse** button.
4. Navigate to the `.chm`, `.hlp`, `.htm`, or `.html` file you want to link to and click the **OK** button. The link to the file appears in the **Help path** text box.

If the **Help path** text box contains a path or symbolic path, also commonly called a relative path, to a `.htm` or `.html` file, LabVIEW ignores the **Help tag** text box. If the **Help path** text box contains a path to a compiled help file (`.chm` or `.hlp`), use the **Help tag** text box in step 5 to link the VI to a specific topic in that help file.

5. (Optional) Enter a `.htm` or `.html` filename or index keyword in the **Help tag** text box if the **Help path** text box links to a compiled help file. If you want to specify a topic in the compiled help file, the topic `.htm` or `.html` filename must match a name of an individual HTML file in the compiled help file. If you want to specify an index entry, the index keyword must match a keyword in the index of the compiled help file.
6. Click the **OK** button to save the changes. When you move the cursor over the VI icon, a blue **Detailed Help** link appears in the **Context Help** window, and the **Detailed Help** button is enabled.

To link VIs to HTML files or compiled help files programmatically, you can use the VI properties `Help:Document Path` and `Help:Document Tag` in a Property Node and the Help functions on the **Help** palette, available on the **Functions** palette when the VI is in a My Computer or Main Application Instance.

## Linking Dialog Box Help Buttons to Compiled Help Files or HTML Files

Dialog boxes with more than one control or indicator usually include a **Help** button that opens a help topic explaining the objects in the dialog box and how to use them.

### Linking to a Topic in a Compiled Help File

You can wire a Boolean button to a Case or Event structure that runs the `_LaunchHelp.vi`, available in the `labview\help` directory.

Wire the name of the `.htm` or `.html` file you want to link to within a compiled help file to the **HTML filename** input of `_LaunchHelp` VI. Wire the name of the compiled help file containing the topic to the **CHM or HLP filename** input.



**Note** You must put the compiled help file in the `labview\help` directory to use the `_LaunchHelp` VI.

### Linking to an Independent HTML File

You also can wire a Boolean button to a Case or Event structure containing the Open URL in Default Browser VI if you want to open the `.htm` or `.html` file in the default browser.



**Note** You must enter the whole path to the `.htm` or `.html` file you want to open.

## Documenting Concepts and Tasks

Conceptual content gives end users background information they might need to use your target. Task-based information gives end users step-by-step instructions about how to use your target in LabVIEW. Most LabVIEW products contain the majority of non-reference content in help. LabVIEW users are accustomed to looking in help for information, and combining VI reference with conceptual and task-based information in a compiled help file allows end users to simultaneously search reference and non-reference content.

While you cannot create non-reference content directly in LabVIEW, many HTML editors and help compilers exist so you can create non-reference content. If you plan on selling integrated target support for LabVIEW, you might consider outsourcing the documentation if you do not have access to technical writers.



**Tip** If your target is an evaluation board, consider including documentation that explains how to create a custom design from the same microcontroller. For example, you might explain how to edit a linker definition file.

## Putting Your CHM Help File in the LabVIEW Help Menu

If you create a compiled help file (.chm) for your target, you can include it in the LabVIEW **Help** menu by installing the .chm file in the `labview\help` directory. If you install the .chm file in the `labview\help` directory, end users can launch your help file by selecting **Help»** `<.chm name>` in LabVIEW.

## Printed and PDF Manuals

---

Include a PDF version of any printed manuals you create. You also can create documents that are available only in PDF format.

## Readme Files

---

You can use readme files for last-minute information and known issues. If you are not reselling LabVIEW and your target, a readme file might be sufficient documentation. LabVIEW uses HTML format for readme files. You also can use text format.

---

# Testing Your Target

You must thoroughly test your target before you can distribute the product to customers or continue with the licensing process described in Chapter 7, *Licensing Your Target and Creating the Installer*. National Instruments does not test and debug your target.

## Manually Testing Your Target

---

Manual tests are an important part of getting your target ready to resell. At a minimum, you must test the following:

- Downloading
- Debugging
- IDE integration
- Elemental I/O
- Driver interfaces and algorithms
- (Optional) Front-panel updating

## Using the Automated Test Framework

---

The Embedded Development Module includes an automated test framework, which is designed to test the run-time library. The test framework has good coverage but cannot test all target-specific code, such as Elemental I/O and interfaces to drivers and algorithms. You must test those features manually. The automated tests verify successful target implementations.

The automatic test framework executes all the tests in a folder you specify in the `LEP_AutoTest.ini` file, located in the `labview\autotest` directory. When you run the automatic tests, LabVIEW creates a folder with a date and time label and saves the test results as text files in that directory.



**Tip** You can specify where LabVIEW creates the results directory in the `LEP_AutoTest.ini` file. By default LabVIEW creates the results directory in the `autotest\results\win32con` directory.

After LabVIEW runs all the tests in the automatic test framework folder and saves the results, LabVIEW creates an `index.html` file that summarizes the results of all the tests. The file lists the test name and whether the test passed or failed. If the test failed, `index.html` provides a hyperlink to the test results text file that reports any errors.



**Note** National Instruments requires that your target pass the automated test suite before continuing with the licensing process.

You can add additional tests to the Embedded Development Module test framework. Refer to the *LabVIEW Embedded Development Module Porting Guide* for more information about the automated test framework.

---

# Licensing Your Target and Creating the Installer

National Instruments must license and sign the target before you can bundle and resell LabVIEW with your target. The steps in the *Process Overview of Bundling Your Embedded Target with LabVIEW* section of Chapter 1, *Distributing Your Embedded Target and LabVIEW*, outline the process National Instruments requires OEMs/VARs to complete to formally create a product for a new embedded target.

After you port LabVIEW to your target, refine the end-user experience, and test your target, you are ready to have National Instruments sign your target and provide you with the necessary files to complete the installer for your product.

## Signing Your Target

---

National Instruments signs your target so end users cannot modify the target-specific plug-in VIs or the functionality of your target.

### Listing the Plug-In VIs for Signing

Use the **Signed Files** tab in the **Target Properties** dialog box in the Target Editor, available by selecting **Tools»Embedded Tools»Target Editor**, to list the files in your target directory that you do not want end users to be able to modify. The Target Editor stores the file list in the `TgtSupp.xml` file in your target directory.



**Tip** Typically, signed files include VIs that launch from a shortcut menu. If you want to prevent end users from modifying VIs that appear on palettes, consider password-protecting these VIs instead of including these VIs in the signed files list. This allows you to make changes to the VIs without resubmitting the `TgtSupp.xml` file to National Instruments. Refer to the *LabVIEW Help* for information about password-protecting VIs.

Refer to the *LabVIEW Help*, available by clicking the **Help** button in the **Target Properties** dialog box, for more information about creating and modifying the `TgtSupp.xml` file.

## Sending Files to National Instruments for Signing and Licensing

You send files to National Instruments for final signing only after you have created, tested, and debugged your target.

When you submit your target to National Instruments for signing and licensing, you must include the following items:

- The `TgtSupp.xml` file for your target
- One  $16 \times 16$  icon in `.ico` format for the National Instruments License Manager
- One  $578 \times 59$  company banner in `.bmp` format
- The URL for your target product
- A short description of the target for end users

Send the files to your designated National Instruments contact person.



**Note** You cannot modify the `TgtSupp.xml` file or signed plug-in VIs after National Instruments signs the `TgtSupp.xml` file.

## What National Instruments Does with Your Target

National Instruments does not review or test your target. You must complete and debug your target before you send the `TgtSupp.xml` file to National Instruments to sign.

National Instruments creates a product DLL and a custom license for your target and bundles the product DLL and the custom license into an installer merge module. National Instruments then sends you the signed `TgtSupp.xml` file, the merge module, and the number of LabVIEW redistributable CD sets that you order.

# Creating an Installer for Your Target

---

When you receive the merge module and signed `TgtSupp.xml` file from National Instruments, your setup developer can create the installation CD for your target.

## Installing the Necessary Components for Your Target

The target installer must include the following:

- All of your target files
- MSI with the product DLL and custom license file
- Signed `TgtSupp.xml` file
- Palettes
- VIs
- Examples
- Documentation
- (Optional) Toolchain for your target



**Note** National Instruments recommends shipping the toolchain that you have tested with your target. As toolchain versions are revised and updated, end users might have difficulty finding the correct version of the toolchain. Also, trying to get a target to work with a different version of a toolchain can introduce hard-to-find errors.

## Locating the LabVIEW 8.5 Directory

The setup developer for your product must create an installer that can dynamically detect where LabVIEW 8.5 is installed on the system in case the end user does not install LabVIEW 8.5 in the default directory.

The LabVIEW installer creates a registry key, **Path**, that stores the path to the main LabVIEW 8.5 directory. The registry location of **Path** is `HKEY_LOCAL_MACHINE\Software\National Instruments\LabVIEW\8.5`.

The setup developer can use the **AppSearch** and **RegLocator** installer database tables to search the registry for the LabVIEW **Path** key and populate a property that specifies the location of the LabVIEW 8.5 directory. The **Directory** table for the target installer can reference this property to install your target files in the LabVIEW 8.5 directory.

## Options for End Users Installing Your Product

---

When you ship your target to customers, you must send them the set of LabVIEW 8.5 redistributable CDs, your target installation CD, and any printed documentation you create for your target. End users can use LabVIEW and your target in evaluation mode or in licensed mode.

As the OEM/VAR, you acquire sets of valid serial numbers from National Instruments to activate the LabVIEW you resell and your target. End users can purchase your target and obtain a valid serial number from you.

### Running LabVIEW in Evaluation Mode

If end users do not activate LabVIEW and your target with the correct serial number, LabVIEW runs in evaluation mode.

National Instruments sets some of the evaluation mode limitations, but you can configure other limitations. Work with your NI contact to define the configurable limitations. Evaluation modes contain the following limitations:

- (Required) Front panel window and block diagram window watermarks on all VIs the end user creates.
- (Required) VIs that the end user runs in a My Computer or main application instance automatically stop running after five minutes.
- (Required) All embedded applications that the end user builds with an evaluation version of LabVIEW contain an *Evaluation version of LabVIEW* message when the application begins running.
- (Configurable) Evaluation stops working after a predetermined amount of time.
- (Configurable) Any applications that the end user builds with the evaluation version stop working after a predetermined date.

### Running LabVIEW in Licensed Mode

When end users activate LabVIEW and your target, LabVIEW runs in licensed mode. End users then can use your target in LabVIEW without any of the limitations listed in the previous section.



---

# Technical Support and Professional Services

As the OEM/VAR, you must consider providing technical support for your product as part of your decision to bundle and resell LabVIEW with your target. Customers will contact you to purchase your target, obtain a valid serial number, and receive technical support for your target. You are the expert for your target, but National Instruments can help you with general support issues.

In addition to the Embedded Development Module documentation, you also have access to the LabVIEW Embedded Target Development Community, the LabVIEW Embedded Discussion Forum, and training.

- The LabVIEW Embedded Target Development Community provides a place for Embedded Development Module users to upload, share, and download plug-in VIs for various hardware platforms. You can access the community by visiting [ni.com/info](http://ni.com/info) and typing `edmcComm`.
- The LabVIEW Embedded Discussion Forum provides a discussion forum for the Embedded Development Module and other LabVIEW Embedded products. You can access the forum by visiting [ni.com/info](http://ni.com/info) and typing `edmForum`.
- On site training at National Instruments corporate headquarters or Web-based training available upon request.

Visit the following sections of the National Instruments Web site at [ni.com](http://ni.com) for technical support and professional services:

- **Support**—Online technical support resources at [ni.com/support](http://ni.com/support) include the following:
  - **Self-Help Resources**—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application

Engineers worldwide in the NI Discussion Forums at [ni.com/forums](http://ni.com/forums). National Instruments Application Engineers make sure every question receives an answer.

For information about other technical support options in your area, visit [ni.com/services](http://ni.com/services) or contact your local office at [ni.com/contact](http://ni.com/contact).

- **Training and Certification**—Visit [ni.com/training](http://ni.com/training) for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit [ni.com/alliance](http://ni.com/alliance).

If you searched [ni.com](http://ni.com) and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of [ni.com/niglobal](http://ni.com/niglobal) to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.