

# Getting Started with the LabVIEW™ Embedded Module for Analog Devices Blackfin Processors

Version 1.0

## Contents

---

Introduction .....	2
Connecting the Blackfin Target to the Host PC.....	2
Launching LabVIEW Embedded Edition and Selecting the Target.....	3
Verifying the Palette View.....	4
Creating the LabVIEW Embedded Project.....	6
Adding a New VI to the Embedded Project.....	8
Creating the Front Panel .....	9
Creating the Block Diagram .....	10
Building the Blackfin VI into a Blackfin Application .....	12
Configuring the Build Options .....	12
Adding Front Panel Update Code (JTAG or USB EZ-KIT Debugging Only).....	15
Configuring the Target and Debugging Options .....	15
Building the Blackfin Application .....	17
Downloading the Blackfin Application .....	17
Debugging the Blackfin Application .....	17
Setting Breakpoints and Probes .....	18
Where to Go from Here .....	20

# Introduction

---

The LabVIEW Embedded Module for Analog Devices Blackfin Processors is a comprehensive graphical development environment for embedded design. Jointly developed by Analog Devices and National Instruments, this module seamlessly integrates the LabVIEW development environment and Blackfin embedded processors.

This module builds on NI LabVIEW Embedded technology, which facilitates dataflow graphical programming for embedded systems and includes hundreds of analysis and signal processing functions, integrated I/O, and an interactive debugging interface. With the Embedded Module for ADI Blackfin, you can enable cache, optimize linking, and view live front panel updates via JTAG, serial, or TCP/IP, as well as use VisualDSP++ compiler options through LabVIEW. The Embedded Module for ADI Blackfin includes the LabVIEW C Code Generator, which generates C code from the LabVIEW block diagram.

Engineers and scientists can achieve faster development times, lower development costs, yet still deliver a high performance embedded processing solution.

Use this tutorial to learn how to create a LabVIEW Embedded Project and build, run, and debug a Blackfin application.

## Connecting the Blackfin Target to the Host PC

---

You must install VisualDSP++ before you can connect the Blackfin target. Refer to the *LabVIEW Embedded Module for Analog Devices Blackfin Processors Release Notes* for installation instructions.



**Note** The Windows Found New Hardware Wizard appears if you are connecting the hardware for the first time.

Complete the following steps to connect the Blackfin target to the host computer.

1. Connect the supplied A/C adaptor to the power connector on the Blackfin target.
2. Connect the supplied USB cable to the USB port on the Blackfin target and to the host computer.

Figure 1 shows the location of the A/C adaptor and USB port on the Blackfin target. Refer to the *ADSP-BF537 EZ-KIT Lite Evaluation System Manual* in the EZ-KIT box for more detailed information about the BF537 EZ-KIT Lite hardware.

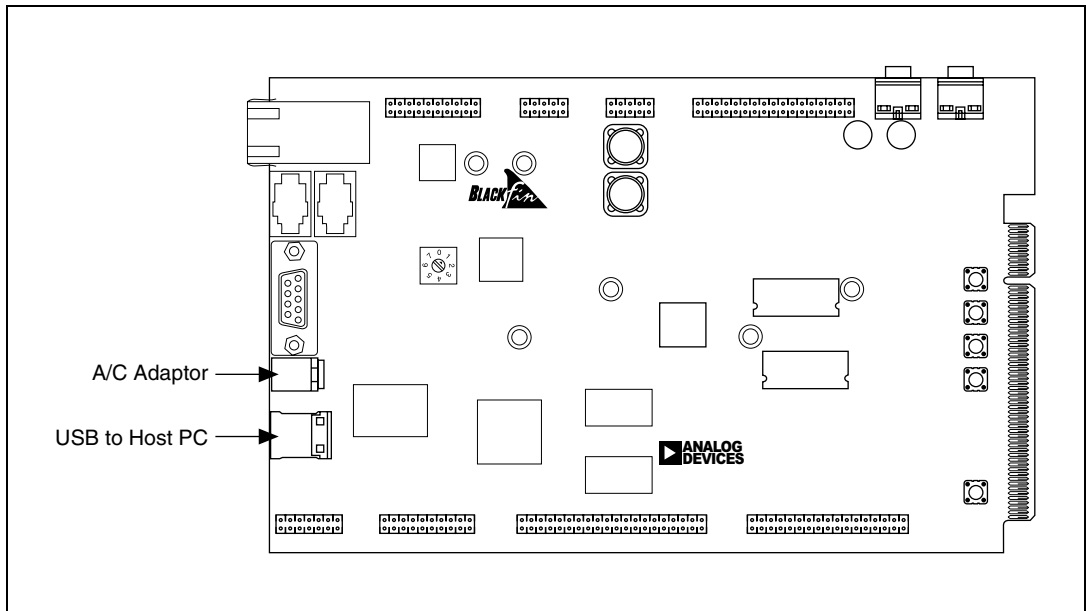


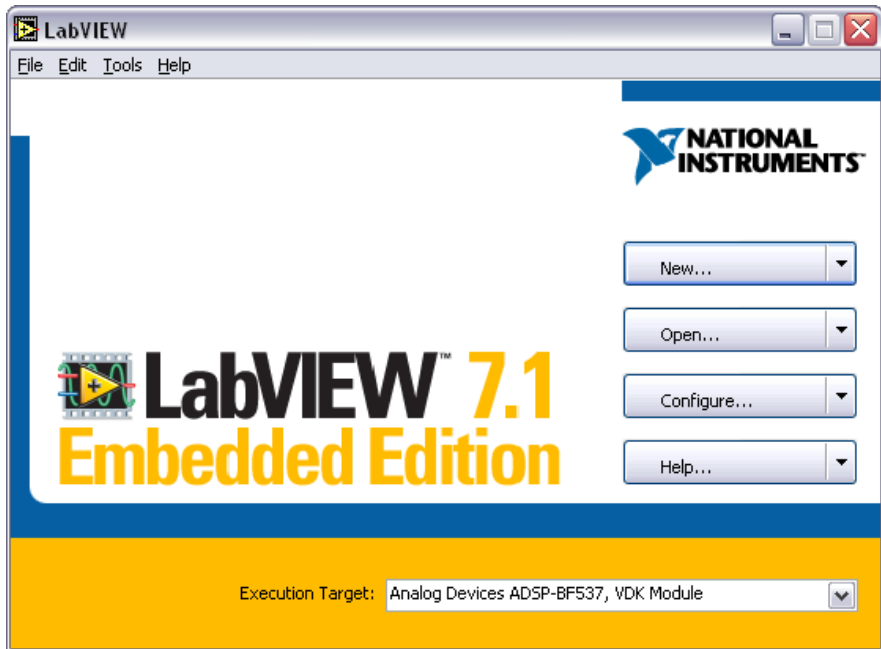
Figure 1. Locating the A/C Adaptor and USB Port on the Blackfin Target

## Launching LabVIEW Embedded Edition and Selecting the Target

The Embedded Module for ADI Blackfin uses LabVIEW 7.1 Embedded Edition, which is a special edition of LabVIEW 7.1 that installs in a separate directory and does not interfere with LabVIEW 7.1 Base, Full, or Professional development systems.

Complete the following steps to launch LabVIEW Embedded Edition and select the Analog Devices Blackfin 537 target.

1. Launch LabVIEW Embedded Edition.
2. In the **LabVIEW** dialog box, shown in Figure 2, select **Analog Devices ADSP-BF537, VDK Module** from the **Execution Target** pull-down menu.



**Figure 2.** Selecting the Execution Target

3. Click the **New** button to open the **New** dialog box, select **Blank VI** from the **Create new** list, and click the **OK** button.



**Tip** You also can click the arrow next to the **New** button and select **Blank VI** from the shortcut menu or press the <Ctrl-N> keys.

The front panel and the block diagram open. The front panel, or user interface, appears with a gray background and includes controls and indicators. The block diagram appears with a white background and includes VIs, functions, and structures that control the front panel objects.

## Verifying the Palette View

The **Controls** palette is available only on the front panel and contains controls and indicators you use to build the user interface. The **Functions** palette is available only on the block diagram and contains the VIs and functions you use to build a VI. The objects on the palettes depend on the current palette view. The Embedded Module for ADI Blackfin includes a Blackfin-specific palette view.

Complete the following steps to verify you are using the Blackfin palette.



**Note** This tutorial verifies the palette view from the **Functions** palette, which opens from the block diagram. You can do the same thing from the **Controls** palette, which opens from the front panel.

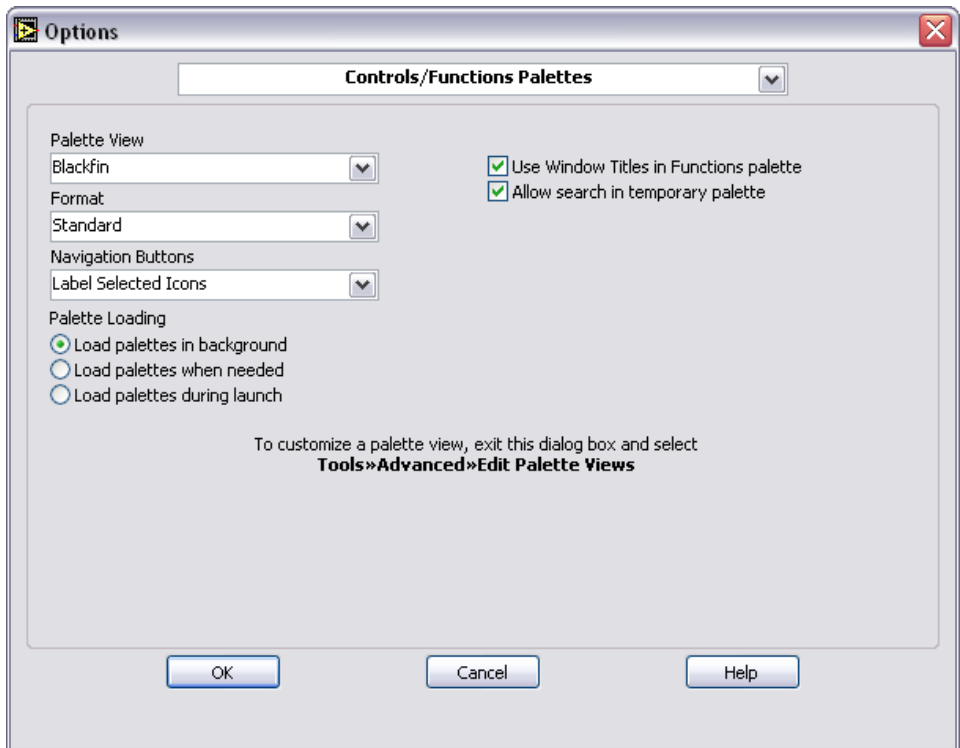
1. If the **Functions** palette is not visible on the block diagram, right-click any blank space on the block diagram to display a temporary version of the **Functions** palette. Click the thumbtack in the upper left corner of the **Functions** palette to pin the palette so it is no longer temporary. You can place the **Functions** palette anywhere on the screen.



**Tip** You also can select **Window»Show Functions Palette** to display the palette.



2. Click the **Options** button, shown at left, on the **Functions** palette to open the **Controls/Functions Palettes** page of the **Options** dialog box.
3. Verify **Blackfin** is the palette view in the **Palette View** pull-down menu as shown in Figure 3.



**Figure 3.** Verifying the Palette View

- Click the **OK** button.

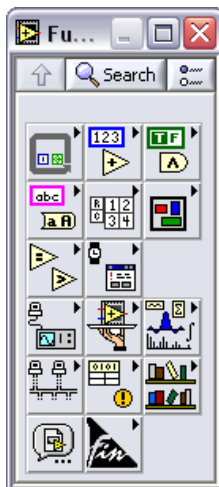


Figure 4. Functions Palette—Blackfin View

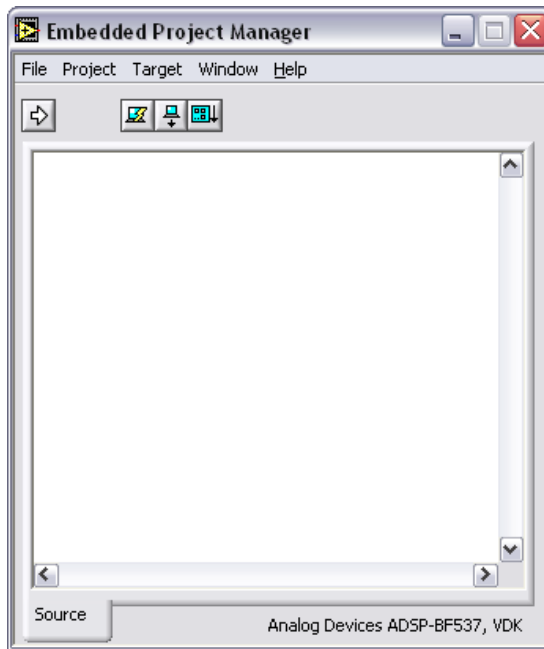
## Creating the LabVIEW Embedded Project

---

Use the **Embedded Project Manager** window to manage groups of VIs, select build options, build, and run Blackfin applications. LabVIEW Embedded Project files have a `.lep` file extension. LEP files contain target-specific build options and other information necessary for the LabVIEW C Code Generator to generate C code from the VIs.

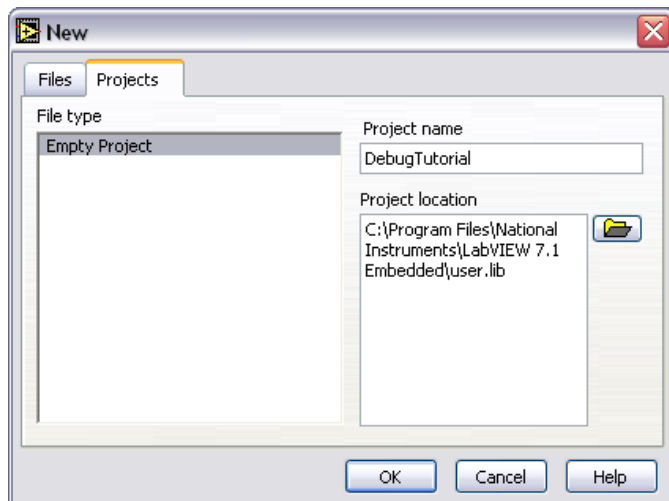
Complete the following steps to create a new Embedded Project.

- Select **Tools»Embedded Project Manager** to open the **Embedded Project Manager** window as shown in Figure 5.



**Figure 5.** LabVIEW Embedded Project Manager Window

2. Select **File»New** in the **Embedded Project Manager** window to open the **New** dialog box as shown in Figure 6.



**Figure 6.** Embedded Project Manager New Dialog Box

3. On the **Projects** tab, enter **DebugTutorial** in the **Project name** text box.

4. Enter a path in the **Project location** path control or browse to the location where you want to save the LEP file.
5. Click the **OK** button.

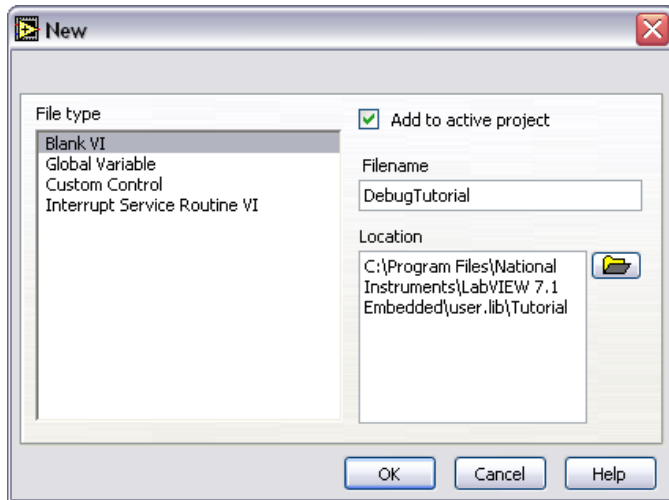
## Adding a New VI to the Embedded Project

---

You can add new or existing VIs to LEP files. You also can add external `.c` files or `.lib` files to LEP files.

Complete the following steps to create a new VI and add it to the embedded project.

1. Select **Project»Add to Project»New** to open the **New** dialog box. You also can right-click **Source Files** and select **Add New** from the shortcut menu.
2. Select **Blank VI** from the **File type** list.
3. Enter `DebugTutorial` in the **Filename** text box as shown in Figure 7.

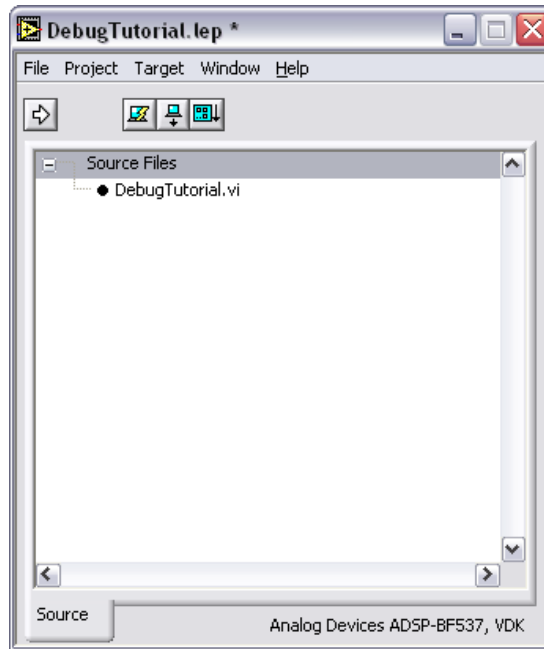


**Figure 7.** Adding a Blank VI to the Embedded Project

4. LabVIEW saves the new VI in the same directory as the LEP file by default. You can save the VI in a different directory, but leave the directory as the default for this tutorial. National Instruments and Analog Devices recommend saving the LEP file and the top-level VI in the same directory.
5. Click the **OK** button.



The front panel of the DebugTutorial VI opens and the DebugTutorial VI appears under **Source Files** in the **Embedded Project Manager** window as shown in Figure 8.



**Figure 8.** Embedded Project Manager Window after Adding a VI

## Creating the Front Panel

---

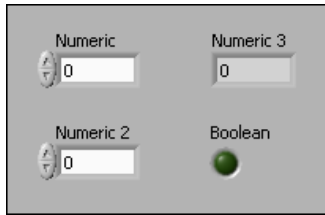
The front panel is the user interface for a VI. You can use the front panel as a debugging interface for Blackfin applications you create with LabVIEW. This tutorial includes an LED indicator that lights if the input exceeds a threshold value you define.

Complete the following steps to create the front panel.

1. Place the following controls on the block diagram as shown in Figure 9:
  - Two numeric controls located on the **Controls»Numeric** palette.
  - One numeric indicator located on the **Controls»Numeric** palette.
  - One round LED located on the **Controls»Boolean** palette.



**Tip** If you cannot find the object you are looking for, click the **Search** button on the **Controls** or **Functions** palette toolbar. Type the name of the object for which you want to search. LabVIEW searches as you type and displays any matches in the search results text box.

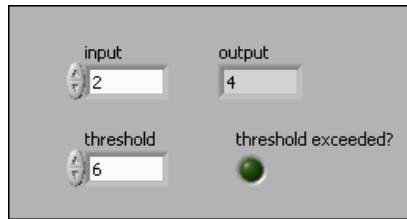


**Figure 9.** Creating the Front Panel

2. Rename the controls by clicking the labels and entering new names as shown in Figure 10.
  - Change one of the numeric controls to **input**.
  - Change the other numeric control to **threshold**.
  - Change the numeric indicator to **output**.
  - Change the round LED to **threshold exceeded?**.



**Tip** Double-click to select a single word in a label. Triple-click to select the entire label.



**Figure 10.** Changing the Labels

3. Change the numeric representation of the numeric controls and the **output** numeric indicator to a 32-bit signed integer by right-clicking each control and indicator and selecting **Representation>I32** from the shortcut menu.

## Creating the Block Diagram

---

The block diagram is the source code for a VI and contains a pictorial description or representation of an application. Wires carry data between the objects, or nodes, on the block diagram. The controls and indicators you added in the *Creating the Front Panel* section appear as terminals on the block diagram.

Complete the following steps to create the block diagram.

1. Switch to the block diagram by clicking the block diagram if it is visible or selecting **Window»Show Block Diagram**.



**Tip** You also can switch to the block diagram by pressing the <Ctrl-E> keys.

2. Select **Help»Show Context Help** to display the **Context Help** window. The **Context Help** window displays basic information about LabVIEW objects when you move the cursor over each object.



**Tip** You also can press <Ctrl-H> to open and close the **Context Help** window.

3. Create the block diagram as shown in Figure 11.



**Tip** You can click the **Search** button on the palette toolbar to perform text-based searches for any control, VI, or function on the **Controls** and **Functions** palettes.



- a. Place a While Loop located on the **Functions»Structures** palette around the controls and indicator on the block diagram. While Loops repeat the subdiagram inside it until the conditional terminal, which is an input terminal, receives a particular Boolean value. Right-click the conditional terminal, shown at left, in the lower right corner of the While Loop and select **Create Constant** from the shortcut menu. The default Boolean constant in the While Loop is FALSE.
- b. Place a Multiply function located on the **Functions»Numeric** palette on the block diagram inside the While Loop.
- c. Wire the **input** control to the **x** input of the Multiply function.
- d. Right-click the **y** input of the Multiply function and select **Create»Constant** from the shortcut menu. Enter 2 to multiply the value of the **input** control by two.
- e. Place a Greater? function located on the **Functions»Comparison** palette on the block diagram.
- f. Wire the **x\*y** output of the Multiply function to the **x** input of the Greater? function.
- g. Wire the **threshold** control to the **y** input of the Greater? function.
- h. Wire the **x > y?** output of the Greater? function to the **threshold exceeded** indicator.
- i. Wire the **output** indicator to the wire connecting the Multiply function and the Greater? function.

- j. Place a Wait Until Next ms Multiple function located on the **Functions»Time, Dialog, & Error** palette inside the While Loop.
- k. Right-click the millisecond multiple input and select **Create»Constant** from the shortcut menu. Enter 100 to wait 100 milliseconds.

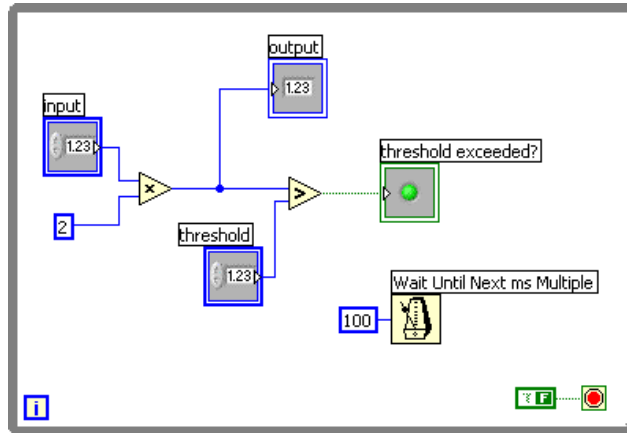


Figure 11. Creating the Block Diagram

4. Save the VI.

## Building the Blackfin VI into a Blackfin Application

Before you can build, or compile, a Blackfin VI into a Blackfin application, you must configure the build options, the target settings, and the debug settings.

### Configuring the Build Options

Build options tell the LabVIEW C Code Generator how to generate the C code and build the Blackfin VI into a Blackfin application.

Complete the following steps to configure the build options for the DebugTutorial VI.

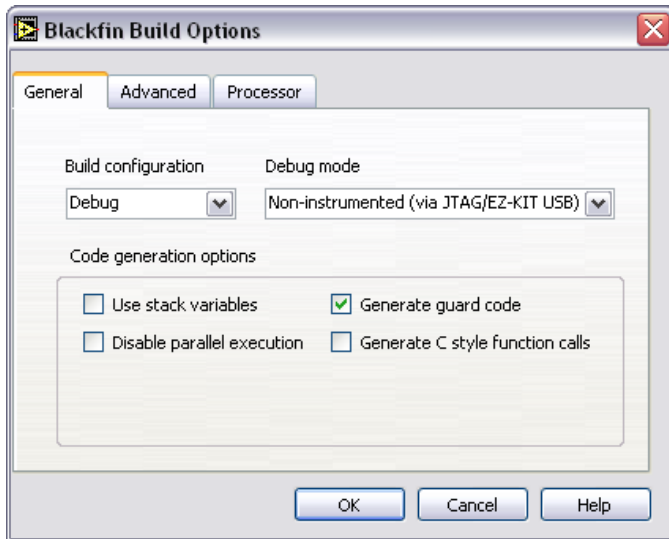
1. Select **Target»Build Options** in the **Embedded Project Manager** window to open the **Blackfin Build Options** dialog box.
2. Select **Debug** in the **Build configuration** pull-down menu. The debug build configuration does not apply any compiler optimizations, which makes the embedded application larger. Use the debug build configuration when you want C source-level debugging.

3. Select a debug mode from the **Debug mode** pull-down menu. You can debug a Blackfin application in the following ways:
  - **Instrumented debugging using a serial port**—Single-stepping and probes are faster than non-instrumented debugging, but using a serial port requires the COM port on the Blackfin target to be connected to the host PC, is more intrusive on real-time performance, and uses a larger amount of memory on the Blackfin target. The LabVIEW C Code Generator adds a communication layer to the generated C code for synchronization and data transfer. You must remove the checkmark from the **Redirect stdout to serial port** checkbox on the **Advanced** tab before you can select the serial port debug option.
  - **Instrumented debugging using a TCP port**—Single-stepping and probes are faster than non-instrumented debugging, but using a TCP port requires that the Blackfin target is connected to an Ethernet port with a DHCP (Dynamic Host Configuration Protocol) server, is more intrusive on real-time performance, and uses a larger amount of memory on the Blackfin target. The LabVIEW C Code Generator adds a communication layer to the generated C code for synchronization and data transfer. You must place a checkmark in the **Enable lwIP TCP/IP support** checkbox on the **Advanced** tab before you can select the TCP port debug option.
  - **Non-instrumented debugging using a JTAG/EZ-KIT USB connection**—Single-stepping and probes are slower than instrumented debugging, but using JTAG or USB does not require a network connection.

This tutorial uses the non-instrumented debug mode as shown in Figure 12, but you can select any debug mode.

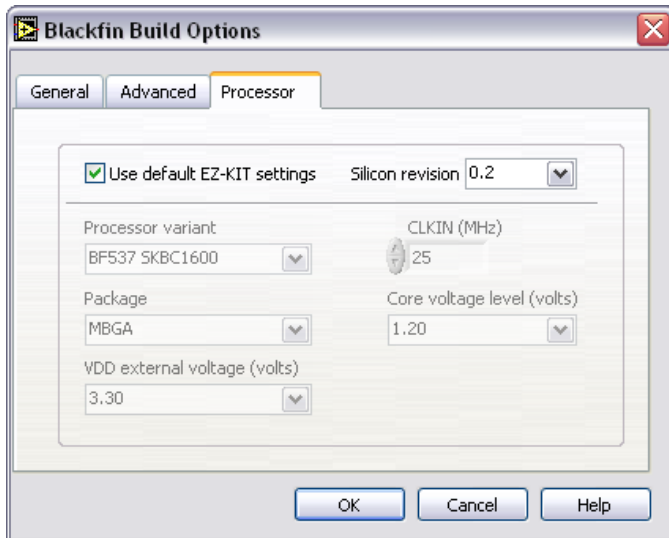


**Note** You must still connect the Blackfin target to the host computer using a JTAG or USB connection to download a Blackfin application to a Blackfin target, run the application on the Blackfin target, reset the processor, and so on, even if you use a serial port or TCP port for debugging.



**Figure 12.** Configuring the Build Options

4. Click the **Processor** tab and verify the silicon revision in the **Silicon revision** drop-down list, shown in Figure 13, matches the silicon revision on the hardware.



**Figure 13.** Verifying the Silicon Revision

5. (Optional) Click the **Help** button to open the *LabVIEW Help* and read a description of each build option.
6. Click the **OK** button.

## Adding Front Panel Update Code (JTAG or USB EZ-KIT Debugging Only)

If you selected **Non-instrumented (via JTAG/EZ-KIT USB)** in step 3 in the *Configuring the Build Options* section, you must add code that updates the front panel on the host computer while the Blackfin application is running on the Blackfin target. To add this code, place a BF Update Front Panel VI located on the **Functions»Blackfin»Miscellaneous** palette inside the While Loop as shown in Figure 14. The BF Update Front Panel VI sets up a Background Telemetry Channel (BTC), which allows data to be transferred directly between the host computer and the Blackfin target. BTC allows LabVIEW to constantly refresh the front panel values by providing direct access to the memory on the Blackfin target. In comparison, serial and TCP/IP debugging uses the Blackfin application you create to communicate with the Blackfin target by adding additional calls to the TCP or serial driver for communication.

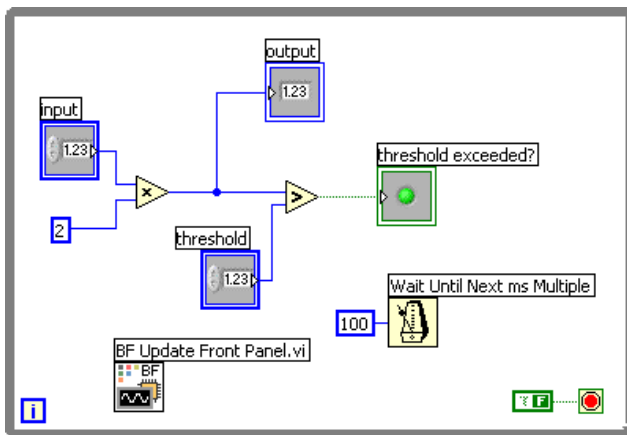


Figure 14. Adding Front Panel Update Code

## Configuring the Target and Debugging Options

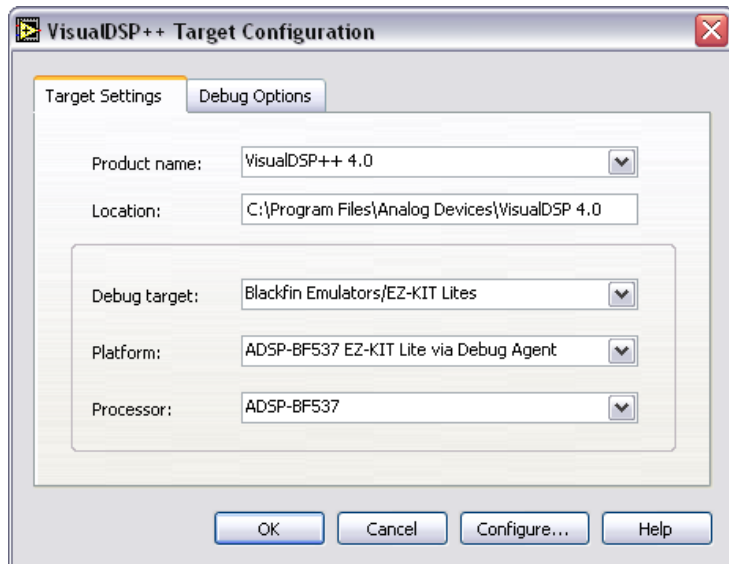
The target options you set tell LabVIEW how the Blackfin target is connected to the host computer. The EZ-KIT Lite is connected to the host computer through the USB port, which also is known as a debug agent.



**Note** You only have to configure the target once unless you change how you connect the target to the host computer or you change silicon revisions.

Complete the following steps to configure the target options.

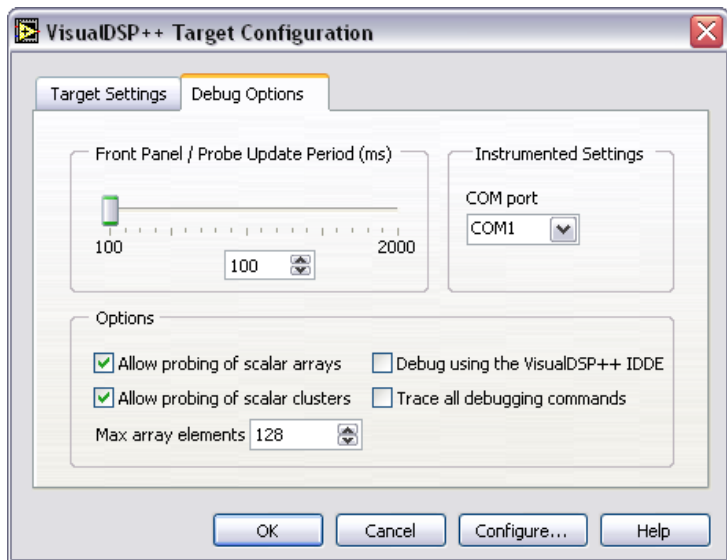
1. Select **Target»Configure Target** in the **Embedded Project Manager** window to open the **VisualDSP++ Target Configuration** dialog box. The current version and location of VisualDSP++ in the **Product name** list and the **Location** text box reference the version of VisualDSP++ on the host computer. Correct the product name and location if necessary.
2. Select **Blackfin Emulators/EZ-KIT Lites** from the **Debug target** pull-down menu.
3. Select **ADSP-BF537 EZ-KIT Lite via Debug Agent** from the **Platform** pull-down menu.
4. Select **ADSP-BF537** from the **Processor** pull-down menu as shown in Figure 15.



**Figure 15.** Configuring the Target Settings

5. Click the **Debug Options** tab to configure the debug options you can use while debugging a Blackfin application on the Blackfin target.
6. Change the **Front Panel/Probe Update Period (ms)** to 100 by moving the slider or typing 100 in the box under the slider as shown in Figure 16, which configures how often the front panel updates with data from the Blackfin application.





**Figure 16.** Configuring the Debugging Options

7. Click the **OK** button.

## Building the Blackfin Application

Select **Target»Build** in the **Embedded Project Manager** window to build the Blackfin VI into a Blackfin application. When you build a Blackfin application, the LabVIEW C Code Generator generates C code from the LabVIEW block diagram using the settings you configure.

## Downloading the Blackfin Application

Select **Target»Download** in the **Embedded Project Manager** window to download the Blackfin application from the host computer to the Blackfin target. This might take a few moments.

## Debugging the Blackfin Application

Complete the following steps to debug the DebugTutorial application.

1. Select **Target»Debug** in the **Embedded Project Manager** window to debug the Blackfin application running on the Blackfin target. The application begins running on the Blackfin target.



**Note** Selecting **Target»Run** in the **Embedded Project Manager** window runs the Blackfin application on the Blackfin target. Any changes you make on the front panel do not have any effect on the Blackfin application running on the Blackfin target. You must

debug the application for interactions with the front panel on the host computer to affect the Blackfin application running on the Blackfin target.

2. Enter a value in the **threshold** front panel numeric control of the DebugTutorial VI on the host computer.
3. Enter different values in the **input** numeric control. In Figure 17, the front panel on the left does not exceed the threshold value. If you enter a number greater than the threshold, the **threshold exceeded?** LED lights as shown in the front panel on the right in Figure 17.

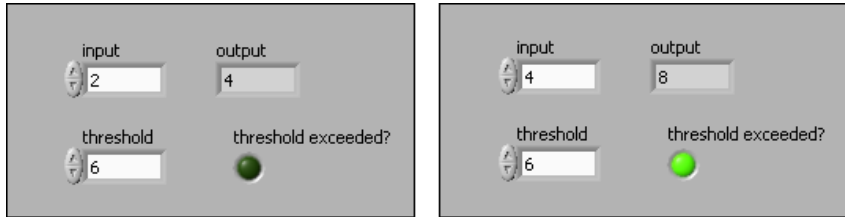


Figure 17. Running the VI



4. Click the **Abort Execution** button, shown at left, to stop the Blackfin application.

## Setting Breakpoints and Probes

Complete the following steps to debug the DebugTutorial application with breakpoints and probes.

1. Switch to the block diagram if it is not visible.
2. Right-click the Multiply function and select **Set Breakpoint** from the shortcut menu. The breakpoint is highlighted with a red border around the function. When you run the Blackfin application, execution pauses just before the function executes. If you are using JTAG or USB/EZ-KIT for debugging, LabVIEW might prompt you to halt the processor.
3. Select **Target»Debug** in the **Embedded Project Manager** window to start running the DebugTutorial application in debug mode. The DebugTutorial is still in memory on the Blackfin target so you do not need to rebuild or redownload the application. LabVIEW prompts you if you need to rebuild or redownload the Blackfin application to the Blackfin target.

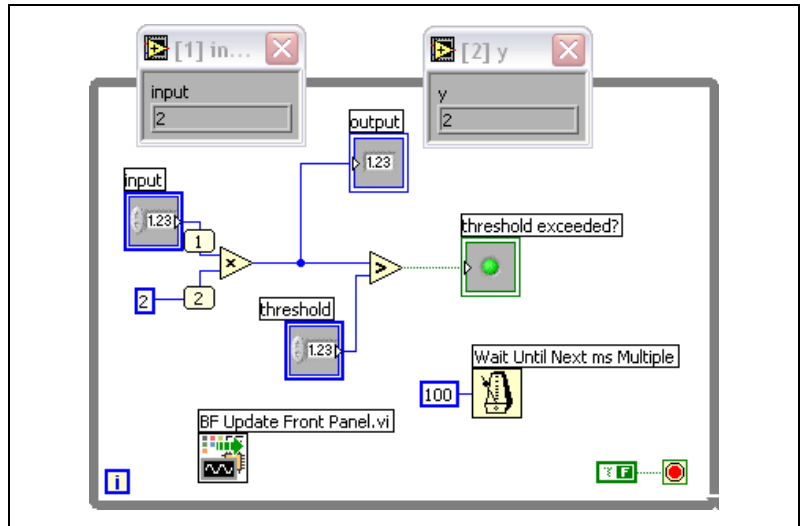
The DebugTutorial application begins running on the Blackfin target. When you reach a breakpoint during execution, the Blackfin target halts all operation, the application pauses, and the **Pause** button, shown at left, appears red and changes to a **Continue** button.



4. Add a probe to see the values on the wire coming into the Multiply function.
  - a. Click the wire coming into the x input.
  - b. Click the wire coming into the y input.

A floating **Probe** window appears after you create each probe.

LabVIEW numbers the **Probe** windows automatically and displays the same number in a glyph on the wire you click as shown in Figure 18.



**Figure 18.** Creating Probes



5. Click the **Continue** button, shown at left, a few times to see the values in the **Probe** windows change as the Blackfin application executes additional iterations of the While Loop.
6. Click the **Step Over** button, shown at left, to execute a node and pause at the next node. The node blinks when it is ready to execute.
7. Continue clicking the **Step Over** button to step through the rest of the block diagram.
8. Click the **Abort Execution** button to stop the application.

# Where to Go from Here

---

National Instruments provides many resources to help you succeed with your NI products. Use the following documentation resources as you start exploring LabVIEW and the Embedded Module for ADI Blackfin.

- *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help** in LabVIEW, provides information about LabVIEW programming, step-by-step instructions for using LabVIEW, and reference information about LabVIEW VIs, functions, palettes, menus, and tools. Refer to the **Embedded Module for ADI Blackfin** book on the **Contents** tab of the *LabVIEW Help* for information specific to the Embedded Module for ADI Blackfin and Blackfin applications.
- Context help provides brief descriptions of VIs and functions with a link to the complete reference for a VI or function. Select **Help» Show Context Help** to open the **Context Help** window.
- Examples are available in the `labview\embedded\examples\lvemb\Blackfin` directory and can help you get started creating Blackfin VIs.
- The readme file, available by selecting **Start»All Programs»National Instruments»LabVIEW 7.1 Embedded Edition»Readme** and opening `readme_BLACKFIN.html`, contains known issues and last-minute information.
- The *LabVIEW Embedded Module for Analog Devices Blackfin Processors Release Notes*, available by selecting **Start»All Programs»National Instruments»LabVIEW 7.1 Embedded Edition»LabVIEW Manuals** and opening `Blackfin_Release_Notes.pdf`, contains system requirements and installation instructions.
- The *Getting Started with ADSP-BF537 EZ-KIT Lite* manual, available in the EZ-KIT box, familiarizes you with the hardware capabilities of the EZ-KIT.
- The *ADSP-BF537 EZ-KIT Lite Evaluation System Manual*, available in the EZ-KIT box, describes the operation and configuration of the board components and provides a schematic for reference.
- The *DAQ Adapter User Guide* describes how to install the National Instruments DAQ Adaptor and provides pinout information. The NI DAQ Adaptor comes in the EZ-KIT box.

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. 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).