Getting Started with the LabVIEW™ Embedded Module for ARM Microcontrollers 1.1

For the Keil MCB2300

The LabVIEW Embedded Module for ARM Microcontrollers is a comprehensive graphical development environment for embedded design. Jointly developed by Keil–An ARM Company and National Instruments, this module seamlessly integrates the LabVIEW graphical development environment and ARM microcontrollers. You can lower development costs and achieve faster development times by using the Embedded Module for ARM Microcontrollers to program ARM targets.

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 ARM Microcontrollers, you can optimize linking and view live front panel updates using JTAG, serial, or TCP/IP. The Embedded Module for ARM Microcontrollers includes the LabVIEW C Code Generator, which generates C code from the LabVIEW block diagram.

This manual includes system requirements, installation instructions, new features, and a step-by-step tutorial that shows you how to build, run, and debug an ARM application.

Contents

NATIONAL INSTRUMENTS™

System Requirements

The Embedded Module for ARM Microcontrollers has the following requirements:

- A computer with Windows Vista/XP/2000
- RealView Microcontroller Development Kit including Keil µVision3
- LabVIEW 8.6 with embedded support
- Keil ULINK2 USB-JTAG adaptor

Refer to the *LabVIEW Release Notes*, available by selecting **Start»All Programs»National Instruments»LabVIEW»LabVIEW Manuals** and opening `LV_Release_Notes.pdf`, for information about LabVIEW development system requirements.

Installing the Embedded Module for ARM Microcontrollers

The Embedded Module for ARM Microcontrollers installer includes LabVIEW 8.6 with embedded support. If you have LabVIEW 8.6 already installed, you can install LabVIEW with embedded support without first uninstalling LabVIEW 8.6. However, you must install the RealView Microcontroller Development Kit before you install the Embedded Module for ARM Microcontrollers.

Complete the following steps to install the RealView Microcontroller Development Kit and the Embedded Module for ARM Microcontrollers.

1. Log in as an administrator or as a user with administrator privileges.
2. Insert the LabVIEW Embedded Module for ARM Microcontrollers installation DVD and select to install the RealView Microcontroller Development Kit.

Tip If the installer does not automatically begin, double-click `MDK_LV.exe` on the DVD to begin installation of the RealView Microcontroller Development Kit.

3. Follow the instructions on the screen for installing the RealView Microcontroller Development Kit.
4. Activate the Keil µVision License ID Code (LIC). Complete the following steps to activate the LIC. Skip this step if you are evaluating the Embedded Module for ARM Microcontrollers. Refer to the *Evaluating the Embedded Module for ARM Microcontrollers* section for more information about running in evaluation mode.
   a. Launch Keil µVision by selecting **Start»All Programs» Keil uVision3**.
   b. Select **File»License Management** to display the **License Management** dialog box.
   c. Click the **Help** button to open the *ARM Development Tools* help file.
   d. Follow the instructions for obtaining a single-user license. You need an internet connection and a product serial number (PSN) to activate the license. The PSN is an alphanumeric value located on the Certificate of Ownership or license card included with purchased products.
   e. After you add the LIC to the **License Management** dialog box, click the **Close** button to close the dialog box.
   f. Exit Keil µVision before installing the Embedded Module for ARM Microcontrollers.

Refer to the Keil Web site at `www.keil.com/license` for more information about activating Keil µVision.

5. If the installer welcome screen is still visible, select to install the Embedded Module for ARM Microcontrollers. If the installer welcome screen is not visible, double-click `setup.exe` on the DVD to begin installation of the Embedded Module for ARM Microcontrollers.
6. Follow the instructions on the screen for installing the Embedded Module for ARM Microcontrollers. The installation DVD installs both LabVIEW with embedded support and the Embedded Module for ARM Microcontrollers.

   (Luminary Micro EK-LM3S8962) You must select the **Custom** installation option and choose to install the **Luminary Micro Driver for EK-LM3S8962**. A software installation alert might appear during the driver installation. Click the **Continue Anyway** button to continue with the installation.
7. Follow the activation instructions that appear on the screen. Skip this step if you are evaluating the Embedded Module for ARM Microcontrollers. Refer to the *Evaluating the Embedded Module for ARM Microcontrollers* section for more information about running in evaluation mode.

   You also can use the NI License Manager, available by selecting **Start»All Programs»National Instruments»NI License Manager**, to activate National Instruments products. Refer to the *National Instruments License Manager Help*, available by selecting **Help» Contents** in the NI License Manager, for more information about activating NI products.
8. Restart the computer when the installer prompts you and log in as an administrator or as a user with administrator privileges.

Evaluating the Embedded Module for ARM Microcontrollers

You can install and evaluate the Embedded Module for ARM Microcontrollers for 60 days. When you run the Embedded Module for ARM Microcontrollers in evaluation mode, LabVIEW includes the following limitations:

Note If you are evaluating the Embedded Module for ARM Microcontrollers with an already licensed and activated LabVIEW development system, these limitations apply only to ARM targets, VIs, and applications.

- **A 60-day time limit until the evaluation version expires**—While not activated, LabVIEW prompts you to activate the product each time you launch LabVIEW. You also receive a warning when you build a VI into an application until you activate the Keil µVision License ID Code (LIC). After the evaluation period for LabVIEW expires, you are no longer able to launch LabVIEW until you purchase and activate the Embedded Module for ARM Microcontrollers.
- **A 128 KB size limit**—Any applications you create and build with LabVIEW and Keil µVision are limited to 128 KB.
- **An evaluation version watermark during the 60-day time limit**—All user VIs and controls contain an evaluation watermark.

Installing the MCB2300 Evaluation Board

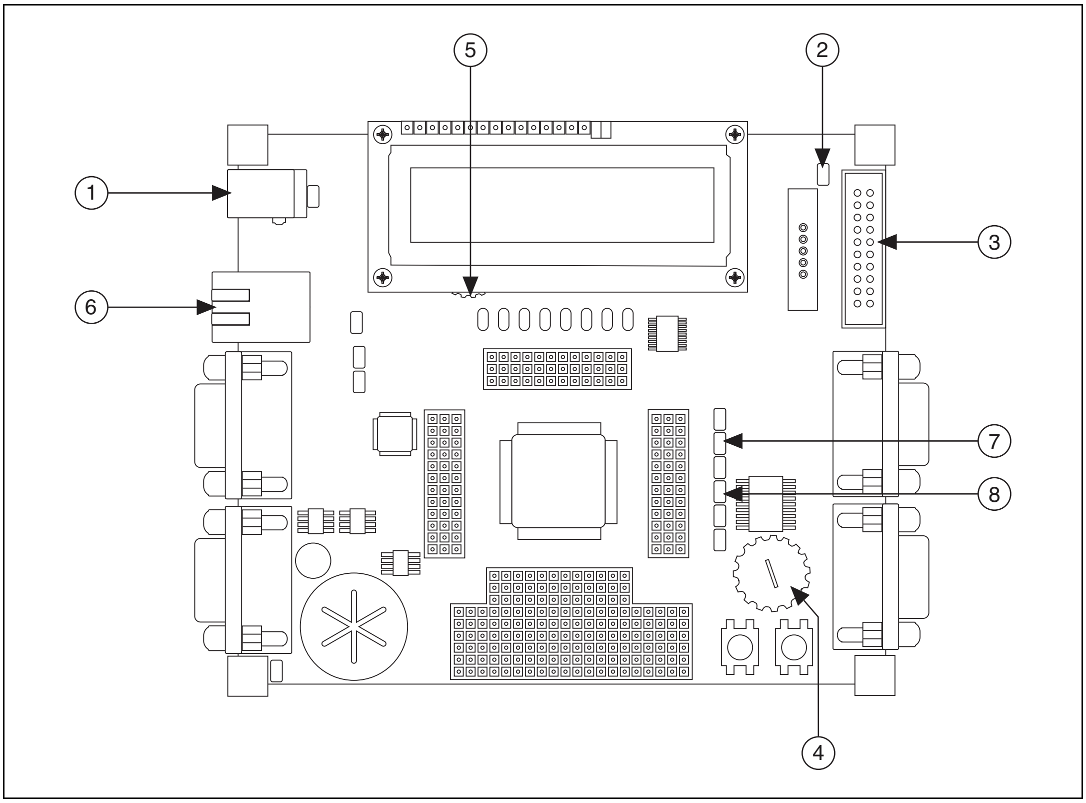
You need the following items to use the MCB2300 evaluation board with JTAG emulation.

- MCB2300 board
- An IBM-compatible PC with two unused USB ports: one to supply power to the MCB2300 board and the other to perform ULINK2 USB-JTAG downloading and debugging
- ULINK2 USB-JTAG adaptor (included)
- Two USB serial cables, each no longer than 10 feet (included)

Refer to the hardware documentation for required accessories such as cables and adaptors.

Caution Be careful when removing the board from the package and handling the board to avoid the discharge of static electricity, which might damage some components.

Figure 1 shows the location of some of the parts on the MCB2300 evaluation board. Refer to the hardware documentation for more information about the evaluation board.



1	USB Connector	4	Potentiometer (Analog Input AD0)	7	ISP Jumper
2	Power LED	5	Contrast Control for LCD	8	RST Jumper
3	JTAG Connector	6	Ethernet Connector		

Figure 1. Locating Parts for the MCB2300 Installation

Complete the following steps to install the board. You do not have to open the computer case to install the board.

1. Verify that you have Keil µVision3 installed. µVision is a part of the RealView Microcontroller Development Kit.

   You can look for the `Keil\uv3` directory on the hard disk or select **Start»All Programs** and locate the shortcut to Keil µVision3. Do not launch µVision3 from the shortcut if you are going to use LabVIEW.

   Refer to the *Installing the Embedded Module for ARM Microcontrollers* section for information about installing the RealView Microcontroller Development Kit.

2. Connect the ULINK2 USB-JTAG adaptor to a USB port on the host computer.

   If this is the first time connecting the ULINK2 USB-JTAG adaptor to the computer, the connection activates the Windows Found New Hardware icon in Windows. A Windows message notifies you when the new device is ready for use and the hardware installation is complete.
3. Connect the ULINK2 USB-JTAG adaptor to the JTAG connector on the board.
4. Connect the USB connector on the board to a USB port on the host computer. This USB connection provides power to the board. The power LED illuminates on the board.

Note The board remembers the last program that ran because you must program the flash memory on the board to run an application. Therefore, the board begins running the last application as soon as the board receives power. You must download a new application to change the start-up behavior of the board.

5. Verify that the RST and ISP jumpers are off if you plan to use the COM0 port. Refer to Figure 1 to locate the RST and ISP jumpers. Refer to the jumper settings configuration topic in the *MCB2300 User's Guide*, available by navigating to `Keil\ARM\Hlp` and opening `mcb2300.chm`, for information about configuring jumpers for other programming utilities, such as Flash Magic.

What's New

The Embedded Module for ARM Microcontrollers includes the following new features:

New Preset Build Specification Configurations

The **Build Specification Properties** dialog box now includes preset build configurations, such as optimizing code generation for application speed or size. When you use one of the preset configurations, LabVIEW sets the appropriate build options automatically. You can override individual options in a preset configuration.

To use a preset build configuration, select a configuration from the **Run-Time Options** pull-down menu on the **Application Information** page of the **Build Specification Properties** dialog box.

Improved Build Specification Conflict Notification

The **Build Specification Properties** dialog box now includes a **Build settings conflicts** area at the bottom of each page. If you select build settings that are incompatible, such as parallel execution and expression folding, you receive notification in the **Build settings conflicts** area.

New VI-Specific Code Generation Options

You now can optimize subVI calls and inline subVIs into callers, which can eliminate overhead and increase code optimization.

- **Optimize subVI calls**—Generates C code for subVI calls with as little default data initialization as possible. You cannot debug a VI with optimized subVI calls.
- **Allow inlining**—Allows inlining of subVIs into callers. Inlining subVIs is most useful for small subVIs, VIs with many calls in a loop, or subVIs with only one call site. The default is **True**. This option only allows inlining. To actually inline a subVI, you must select **True** from the **Inline subVI** pull-down menu on the **Source File Settings** page in the **Build Specification Properties** dialog box.

To set code generation options for a VI, select **File»VI Properties** from the front panel window or block diagram window to open the **VI Properties** dialog box. Select **C Code Generation Options** from the **Category** pull-down menu. You can select **From project**, **True**, or **False**. Select **True** to enable the option and **False** to disable it. The default is **From project**, so you only need to set the options in the **VI Properties** dialog box if you want to override the code generation settings in the project.

New Project-Level Code Generation Options

You now can allocate constants for arrays, clusters, strings, variants, and waveforms with build options on the **Application Information** page in the **Build Specification Properties** dialog box.

Using stack variables overrides constant allocation. If you place a checkmark in the **Use stack variables** checkbox, **Allocate constants** is always **First use** and **Deallocate constants** is always **Out of scope**. **First use** and **Out of scope** was the default behavior in previous versions.

- **Allocate constants—**Specifies when LabVIEW allocates memory for constants.
  - **First use**—Allocates memory the first time you use constants on the block diagram.
  - **Containing loop—**Allocates memory outside of the loop that contains constants.

    - **VI initialization**—Allocates memory when the VI that contains the constants is called.
    - **Application initialization**—Allocates memory when the built application begins running on the target.
- **Deallocate constants**—Specifies when LabVIEW frees memory resources for constants.
    - **Out of scope**—Frees memory resources when the constants are no longer used.
    - **VI end**—Frees memory resources when a VI containing constants finishes executing.
    - **Application end**—Frees memory resources when the built application finishes executing on the target.

Shared Variable Support

You can use single-process and network-published shared variables to share data among VIs in the same embedded application or to read data from and write data to other network-published shared variables. Embedded targets support single-process shared variables and network-published shared variables. Embedded targets do not support hosting shared variables.

Use the Embedded Variable Connection Manager, available by selecting **Tools»ARM Module»Embedded Variable Connection Manager**, to start a TCP/IP connection service between a host computer and embedded target for reading and writing shared variables.

Fixed-Point Support

The fixed-point data type has limited support.

Note Overflow mode is supported, but overflow status is not supported.

Supported Numeric Functions

The following Numeric functions support the fixed-point data type:

- Absolute Value
- Add
- Decrement
- Increment
- Multiply
- Negate
- Round To Nearest

- Round Toward +Infinity
- Round Toward –Infinity
- Scale By Power Of 2 Function
- Sign
- Subtract
- Square

Comparison Functions

The following Comparison functions support the fixed-point data type:

- Equal?
- Equal To 0?
- Greater Or Equal?
- Greater Or Equal To 0?
- Greater?
- Greater Than 0?
- Less Or Equal?
- Less Or Equal To 0?
- Less?
- Less Than 0?
- Not Equal?
- Not Equal To 0?

Conversion Functions

The following Conversion functions support the fixed-point data type:

- Boolean Array To Number
- Number To Boolean Array
- To Byte Integer
- To Double Precision Float
- To Extended Precision Float
- To Fixed-Point
- To Long Integer
- To Quad Integer
- To Single Precision Float
- To Unsigned Byte Integer
- To Unsigned Long Integer

- To Unsigned Quad Integer
- To Unsigned Word Integer
- To Word Integer

Data Manipulation Functions

The following Data Manipulation functions support the fixed-point data type:

- Flatten To String
- Logical Shift
- Rotate Left With Carry
- Rotate Right With Carry
- Type Cast
- Unflatten From String

String/Number Conversion Functions

The following String/Number Conversion functions support the fixed-point data type:

- Decimal String To Number
- Fract/Exp String To Number
- Hexadecimal String To Number
- Number To Decimal String
- Number To Engineering String
- Number To Exponential String
- Number To Fractional String
- Number To Hexadecimal String
- Number To Octal String
- Octal String To Number

New Target Support

In addition to the Keil MCB2300 and the Luminary Micro EK-LM3S8962 boards, the Embedded Module for ARM Microprocessors now also supports the Keil MCB2400 board, which contains an NXP LPC2468 microcontroller. Refer to the Keil Web site at `www.keil.com` for more information about or to purchase the MCB2400 board.

New VIs and Functions

New CAN VIs

The Embedded Module for ARM Microcontrollers now includes a **CAN** palette, which includes the following VIs:

- ARM CAN Open
- ARM CAN Start
- ARM CAN Set Receive ID
- ARM CAN Read
- ARM CAN Write

Use these VIs to communicate with a controller area network (CAN).

New Memory Access VIs

The Embedded Module for ARM Microcontrollers now includes a **Memory Access** palette, which includes the following VIs:

- CCG Mem Peek 8
- CCG Mem Peek 16
- CCG Mem Peek 32
- CCG Mem Poke 8
- CCG Mem Poke 16
- CCG Mem Poke 32

Use these VIs to read and write values to specific memory addresses.

New Console Output VI

The Embedded Module for ARM Microcontrollers now includes a CCG Console Output VI for `printf` functionality. Use this VI to print text to the standard output stream, stdout, of the operating system on the target. The ARM Console Output VI in version 1.0 has been deprecated.

In Place Element Structure Support

The Embedded Module for ARM Microcontrollers now supports the In Place Element structure, which controls how the LabVIEW compiler performs certain operations and, in some cases, increases memory and VI efficiency.

Synchronization Functions Support

The Embedded Module for ARM Microcontrollers supports the following new Synchronization functions:

- Lossy Enqueue Element
- Wait on Notification from Multiple with Notifier History
- Wait on Notification with Notifier History

Tutorial for the Embedded Module for ARM Microcontrollers

Use this tutorial to learn how to build, run, and debug an ARM application. In this tutorial, you create a VI that you build into an application and run on the ARM target. You use the front panel on the host computer as a debugging interface between the host computer and the target. An LED indicator on the front panel lights when an input exceeds a threshold you define. Then, you add Elemental I/O to the VI that lights an LED on the target when the input exceeds the threshold.

Creating the LabVIEW Project

Use LabVIEW projects (`.lvproj`) to group together LabVIEW files and non-LabVIEW files, create build specifications for building a VI into an ARM application, and run the application on the target. You must use a project to build an ARM VI into an ARM application.

Complete the following steps to create a project with an ARM target and a blank VI.

1. Launch LabVIEW. In the **Getting Started** window, select **ARM Project** from the **Targets** pull-down menu. Click the **Go** button to launch the ARM Project Wizard.
2. Select **New ARM project, blank VI** in the **Project type** pull-down menu to create the LabVIEW project with a blank VI.

Tip The **New ARM project, existing VI** imports an existing VI rather than creating a new, blank VI.

3. Click the **Next** button to display the **Select ARM target type** page.
4. Select the target from the **Target type** pull-down menu.
5. Click the **Next** button to display the **System preview** page.

6. Verify the **Create a build specification** checkbox contains a checkmark. Build specifications contain the build settings and code generation options to use when you build a VI into an application.
7. Click the **Finish** button.

   Because the **Create a build specification** checkbox contains a checkmark, the ARM Project Wizard creates a build specification with default settings. LabVIEW prompts you to save the project and VI before the ARM Project Wizard can create the build specification.

   The project now appears in the **Project Explorer** window.

Reviewing the Project Explorer Window

The **Project Explorer** window includes two pages, the **Items** page and the **Files** page. The **Items** page displays the project items as they exist in the project tree. The **Files** page displays the project items that have a corresponding file on disk. Project operations on the **Files** page both reflect and update the contents on disk. You can switch from one page to the other by clicking the **Items** and **Files** tabs or by right-clicking a folder or item under a target and selecting **Show in Items View** or **Show in Files View** from the shortcut menu.

Expand the ARM target in the **Project Explorer** window. The VI you created with the ARM Project Wizard appears under the ARM target. LabVIEW automatically adds **Dependencies** and **Build Specifications** under the target. SubVIs appear under **Dependencies** when you add a VI that contains subVIs to a project. Build specifications you create appear under **Build Specifications**.

To see the build specification you created with the ARM Project Wizard, expand the **Build Specifications** item under the ARM target in the **Project Explorer** window. **Application** is the default build specification name. You can rename the build specification by right-clicking **Application** and selecting **Rename** from the shortcut menu or by double-clicking the build specification, which opens the **Build Specification Properties** dialog box, and entering a new name in the **Build specification name** text box. Refer to the *Verifying the Build Specification* section for more information about the **Build Specification Properties** dialog box.

Creating the Front Panel

The front panel window usually contains the user interface for a VI. ARM applications do not include a user interface, but you can use the front panel window as a debugging interface. In this tutorial, you create a VI with an LED indicator that lights on the front panel if the input exceeds a threshold value you define.

Complete the following steps to create the front panel debugging interface.

1. Add the following controls to the front panel window:
   - Two numeric controls located on the **Numeric** palette.
   - One numeric indicator located on the **Numeric** palette.
   - One round LED located on the **Boolean** palette.

Tip If you cannot find the object you want, click the **Search** button on the **Controls** palette for front panel objects or the **Functions** palette for block diagram objects. 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. You also can press the <Ctrl-Space> keys or select **View»Quick Drop** to display the **Quick Drop** dialog box. Type the name of the object you want to add to the front panel or block diagram windows.

2. Rename the controls as shown in Figure 2.
   - Rename one of the numeric controls to **input**.
   - Rename the other numeric control to **threshold**.
   - Rename the numeric indicator to **output**.
   - Rename the round LED to **threshold exceeded?**.

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

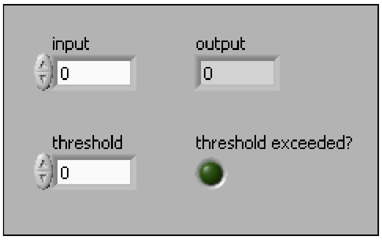


Figure 2. Changing the Labels

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 as shown in Figure 3. This block diagram multiplies an input value by 2 and then lights an LED if the product is greater than the threshold value you specify.

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 the <Ctrl-H> keys to open and close the **Context Help** window.

3. Place a While Loop, located on the **Structures** palette, around the controls and indicator on the block diagram. While Loops repeat the inner subdiagram 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.
4. Place a Multiply function, located on the **Numeric** palette, on the block diagram inside the While Loop.
5. Wire the **input** control to the **x** input of the Multiply function.
6. 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.
7. Place a Greater? function, located on the **Comparison** palette, on the block diagram.
8. Wire the **x*y** output of the Multiply function to the **x** input of the Greater? function.
9. Wire the **threshold** control to the **y** input of the Greater? function.
10. Wire the **x > y?** output of the Greater? function to the **threshold exceeded** indicator.

11. Wire the **output** indicator to the wire connecting the Multiply function and the Greater? function.
12. Place a Wait Until Next ms Multiple function, located on the **Time, Dialog & Error** palette, inside the While Loop.
13. Right-click the **millisecond multiple** input of the Wait Until Next ms Multiple function and select **Create»Constant** from the shortcut menu. Enter `100` to wait 100 milliseconds between loop iterations.
14. Save the VI.

The block diagram should look similar to Figure 3.

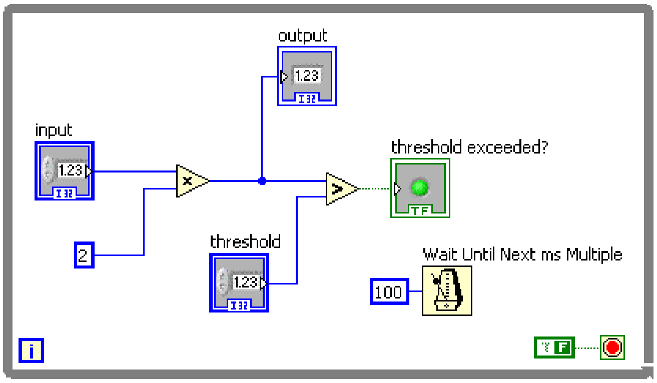


Figure 3. Creating the Block Diagram

Verifying the Build Specification

Use build specifications to specify how the LabVIEW C Code Generator generates C code and how to build the ARM VI into an application.

You can have multiple build specifications for the same target. For example, you might want one build specification that generates debugging information and another build specification that does not generate this extra information. By default, ARM build specifications enable debugging.

Complete the following steps to verify the settings in the build specification you created with the ARM Project Wizard.

1. Right-click the build specification in the **Project Explorer** window and select **Properties** from the shortcut menu to display the **Build Specification Properties** dialog box.

Tip You also can double-click the build specification to open the **Build Specification Properties** dialog box.

2. Verify that the **Enable debugging** checkbox contains a checkmark and the current debugging mode is JTAG.

Tip The current debugging mode is shown under the **Enable debugging** checkbox. You select the debugging mode on the **Advanced Debugging Options** page.

3. Verify the execution location is **Run on target using ULINK2** to run the application on the evaluation board.
4. Select the **Source Files** category and verify that the VI is in the **Top-level VI** text box. When you use the ARM Project Wizard to create a project, LabVIEW automatically uses the VI the wizard creates as the top-level VI. When you create a project without using the wizard, you must manually select the top-level VI by clicking the blue right arrow button, shown at left, to move a VI from the source files list to the **Top-level VI** text box. If the ARM project contains other files, such as `.c` and `.lib` files, add these files to the **Additional files** list.
5. Click the **OK** button to close the dialog box.
6. Select **File»Save All** in the **Project Explorer** window or VI.

Building and Running the ARM Application

After you develop the ARM VI on the host computer, you build the VI into an application that runs on an ARM target. When you build an ARM application, the LabVIEW C Code Generator generates C code from the LabVIEW block diagram using the settings you configure in the **Build Specification Properties** dialog box.

Note You must activate the Keil µVision License ID Code (LIC) before you can build an ARM application with LabVIEW. If the LIC is not activated, you receive a warning when you try to build the application. Refer to the *Activating the Keil µVision License ID Code Readme*, available by selecting **Start»All Programs»National Instruments» LabVIEW»Readme** and opening `readme_ARM_uVision_Licensing.html`, for information about activating the LIC. Refer to the *Evaluating the Embedded Module for ARM Microcontrollers* section for more information about evaluation mode.

Complete the following steps to build, download, and run an ARM application.

1. Click the **Run** button, shown at left, in the VI or right-click the build specification in the **Project Explorer** window and select **Run** from the shortcut menu to build, download, and run the application on the ARM target using the settings in the **Build Specification** dialog box. LabVIEW displays the status of the building process, which includes compiling and linking. In addition, the **Processor Status** window assists in monitoring the download, connection, and execution progress of the application.

   Because the build specification for this application enables debugging, the application runs on the target with front panel updates on the host computer.

Note Click the **OK** button if a dialog box appears notifying you about an updated µVision template.

2. Enter a value in the **threshold** numeric control on the host computer.
3. Enter different values in the **input** numeric control. In Figure 4, the **output** value on the left does not exceed the **threshold** value. If you change the **input** value so that the **output** value is greater than the **threshold** value, the **threshold exceeded?** LED lights.

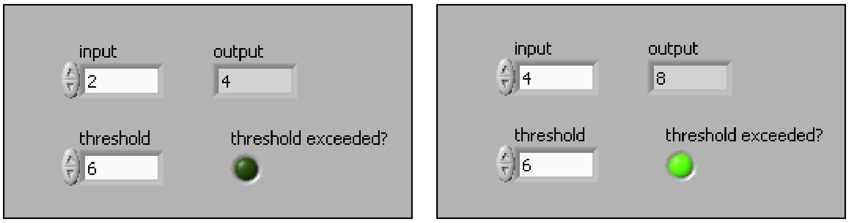


Figure 4. LED Lights when Output Exceeds Threshold

Tip LabVIEW uses default values for controls and indicators when building an ARM VI into an ARM application. To change the initial values, enter the new values in the front panel controls and then select **Edit»Make Current Values Default** to change the initial values. You must rebuild the ARM application if you change the initial values for any of the controls.

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

Debugging with Breakpoints and Probes

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

1. Switch to the block diagram if it is not visible.
2. Right-click the Multiply function and select **Breakpoint»Set Breakpoint** from the shortcut menu. The breakpoint is highlighted with a red border around the function. This breakpoint specifies to pause execution just before the function executes.
3. Click the **Run** button or right-click the build specification in the **Project Explorer** window and select **Debug** or **Run** from the shortcut menu. Save the VI if prompted. LabVIEW also prompts you if you need to rebuild or redownload the ARM application to the ARM target.

   The ARM tutorial application begins running on the ARM target. When the application reaches the breakpoint during execution, the ARM target halts all operation, the application pauses, and the **Pause** button on the host computer, shown at left, appears red and changes to a **Continue** button.

4. Add probes on the wires coming into the Multiply function to see the values.
   a. Click the wire coming into the **x** input.
   b. Click the wire coming into the **y** input.

   As shown in Figure 5, 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.

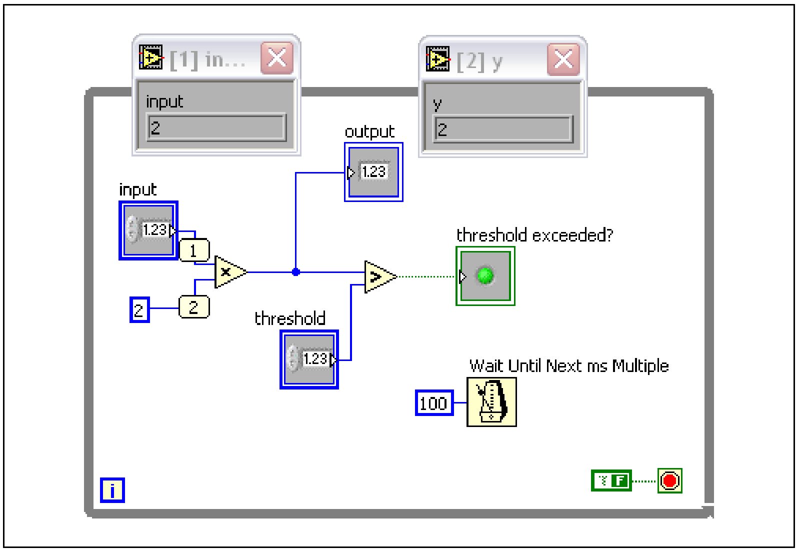


Figure 5. Creating Probes

5. Enter a different value in the **input** numeric control and click the **Continue** button, shown at left, to see the value in the first **Probe** window change as the ARM application executes additional iterations of the While Loop. Repeat entering a different value in the **input** numeric control and clicking the **Continue** button a few times.
6. Click the **Step Over** button, shown at left, to execute the Multiply function and pause at the Greater? function, which 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.
9. Right-click the Multiply function and select **Breakpoint»Clear Breakpoint** from the shortcut menu to remove the breakpoint.

Using Elemental I/O

Elemental I/O resources are fixed elements of ARM targets that you use to transfer data among the different parts of the target. Each Elemental I/O resource has a specific type, such as digital, analog, or PWM. For example, you can use digital Elemental I/O resources to manipulate an LED on the ARM target. Refer to the *LabVIEW Help* for more information about using Elemental I/O with ARM targets.

The following sections describe how to use Elemental I/O to light an LED on the ARM target when the threshold is exceeded.

Adding Elemental I/O Items to the Project

You must add Elemental I/O items to the project before you can use Elemental I/O in an ARM VI. Complete the following steps to add Elemental I/O items to the project.

1. Right-click the target in the **Project Explorer** window and select **New»Elemental I/O** from the shortcut menu to display the **New Elemental I/O** dialog box.
2. Expand **Digital Output** in the **Available Resources** list.
3. Hold down the <Ctrl> key and select **LED1** and **LED2**.
4. Click the **Add** button to add the LED1 and LED2 resources to the **New Elemental I/O** list.
5. Click the **OK** button to add the Elemental I/O items to the LabVIEW project.

   Many pins on ARM targets can have multiple configurations. For example, LED1 and PWM2 both use the same pin on the MCB2300 board. Therefore, you cannot use both LED1 and PWM2 in the same application. If you add LED1 and PWM2 to the project at the same time, LabVIEW indicates a conflict on the PWM2 item in the **Project Explorer** window.

   After you add Elemental I/O items to the project, LabVIEW filters the available resources in the **New Elemental I/O** dialog box to remove resources with pin conflicts. If you right-click the target and select **New»Elemental I/O** from the shortcut menu again, notice that PWM2 is no longer available in the **Available Resources** list because you already added LED1 to the project.

Using Elemental I/O on the Block Diagram

You can use Elemental I/O on the block diagram after you add Elemental I/O items to the project. Complete the following steps to use Elemental I/O on the block diagram.

1. Drag LED1 from the **Project Explorer** window to the block diagram above the **threshold exceeded?** indicator and inside the While Loop.
2. Expand the Elemental I/O Node by dragging the bottom handle until you see LED1 and LED2.
3. Wire the **x > y?** output of the Greater? function to the LED1 and LED2 items in the Elemental I/O Node.

   Refer to the *Using Elemental I/O Nodes* topic in the *LabVIEW Help* for more information about using Elemental I/O Nodes.
4. Right-click the wire that connects the **x > y?** output to LED2 and select **Insert»Boolean palette»Not** from the shortcut menu to place a Not function on the wire. Using the Not function specifies that LED1 and LED2 alternate status such that when LED1 is off, LED2 is on.

Building and Running the Application with Elemental I/O

Note Before you run the application, verify that LabVIEW is downloading to the target and not to the simulator. In the **Build Specification Properties** dialog box, verify **Run on target using ULINK2** is selected.

Complete the following steps to run the ARM application and use Elemental I/O to light an LED on the target.

1. Click the **Run** button. Save the VI if prompted.
2. Click the **Yes** button when LabVIEW prompts you to rebuild the embedded application.
3. Enter different values in the **input** numeric control until the **threshold exceeded?** indicator lights on the front panel. When the **threshold exceeded?** indicator lights on the host computer, LED1 on the ARM target also lights and LED2 turns off.
4. Click the **Abort Execution** button to stop the ARM application.

Where to Go from Here

National Instruments provides many resources to help you succeed with your NI products. Use the following related documentation as you continue exploring LabVIEW and the Embedded Module for ARM Microcontrollers.

- *LabVIEW Help*, available by selecting **Help»Search the LabVIEW 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 ARM Microcontrollers** book on the **Contents** tab of the *LabVIEW Help* for information specific to the Embedded Module for ARM Microcontrollers and the applications you create. The *LabVIEW Help* uses (ARM) in the index to indicate topics specific to the Embedded Module for ARM Microprocessors. The *LabVIEW Help* uses (Embedded Targets) in the index to indicate topics that are relevant to all embedded targets.
- Context help provides brief descriptions of VIs, functions, and dialog boxes. Context help for most VIs and functions include a link to the complete reference for a VI or function. Select **Help»Show Context Help** to display the **Context Help** window.
- Examples, available from the NI Example Finder and in the `labview\examples\lvemb\ARM` directory, can help you get started creating applications.
- The readme file, available by selecting **Start»All Programs»National Instruments»LabVIEW»Readme** and opening `readme_ARM.html`, contains known issues and last-minute information.
- *Getting Started with LabVIEW* manual, available by selecting **Start»All Programs»National Instruments»LabVIEW» LabVIEW Manuals** and opening `LV_Getting_Started.pdf`, provides information about the LabVIEW graphical programming environment and the basic LabVIEW features you use to build data acquisition and instrument control applications.
- The *MCB2300 User's Guide*, available by navigating to the `Keil\ARM\Hlp` directory and opening `mcb2300.chm`, provides information about the MCB2300 evaluation board.

374930B-01 Sep08