

# NI Vision

NI Vision Builder for Automated Inspection  
Development Toolkit Tutorial

## Worldwide Technical Support and Product Information

[ni.com](http://ni.com)

## Worldwide Offices

Visit [ni.com/niglobal](http://ni.com/niglobal) to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

## National Instruments Corporate Headquarters

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

For further support information, refer to the [NI Services](#) appendix. To comment on National Instruments documentation, refer to the National Instruments website at [ni.com/info](http://ni.com/info) and enter the Info Code `feedback`.

# Legal Information

---

## Limited Warranty

This document is provided 'as is' and is subject to being changed, without notice, in future editions. For the latest version, refer to [ni.com/manuals](http://ni.com/manuals). NI reviews this document carefully for technical accuracy; however, NI MAKES NO EXPRESS OR IMPLIED WARRANTIES AS TO THE ACCURACY OF THE INFORMATION CONTAINED HEREIN AND SHALL NOT BE LIABLE FOR ANY ERRORS.

NI warrants that its hardware products will be free of defects in materials and workmanship that cause the product to fail to substantially conform to the applicable NI published specifications for one (1) year from the date of invoice.

For a period of ninety (90) days from the date of invoice, NI warrants that (i) its software products will perform substantially in accordance with the applicable documentation provided with the software and (ii) the software media will be free from defects in materials and workmanship.

If NI receives notice of a defect or non-conformance during the applicable warranty period, NI will, in its discretion: (i) repair or replace the affected product, or (ii) refund the fees paid for the affected product. Repaired or replaced Hardware will be warranted for the remainder of the original warranty period or ninety (90) days, whichever is longer. If NI elects to repair or replace the product, NI may use new or refurbished parts or products that are equivalent to new in performance and reliability and are at least functionally equivalent to the original part or product.

You must obtain an RMA number from NI before returning any product to NI. NI reserves the right to charge a fee for examining and testing Hardware not covered by the Limited Warranty.

This Limited Warranty does not apply if the defect of the product resulted from improper or inadequate maintenance, installation, repair, or calibration (performed by a party other than NI); unauthorized modification; improper environment; use of an improper hardware or software key; improper use or operation outside of the specification for the product; improper voltages; accident, abuse, or neglect; or a hazard such as lightning, flood, or other act of nature.

THE REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND THE CUSTOMER'S SOLE REMEDIES, AND SHALL APPLY EVEN IF SUCH REMEDIES FAIL OF THEIR ESSENTIAL PURPOSE.

EXCEPT AS EXPRESSLY SET FORTH HEREIN, PRODUCTS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND AND NI DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, WITH RESPECT TO THE PRODUCTS, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, AND ANY WARRANTIES THAT MAY ARISE FROM USAGE OF TRADE OR COURSE OF DEALING. NI DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF OR THE RESULTS OF THE USE OF THE PRODUCTS IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NI DOES NOT WARRANT THAT THE OPERATION OF THE PRODUCTS WILL BE UNINTERRUPTED OR ERROR FREE.

In the event that you and NI have a separate signed written agreement with warranty terms covering the products, then the warranty terms in the separate agreement shall 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.

## End-User License Agreements and Third-Party Legal Notices

You can find end-user license agreements (EULAs) and third-party legal notices in the following locations:

- Notices are located in the <National Instruments>\\_Legal Information and <National Instruments> directories.
- EULAs are located in the <National Instruments>\Shared\MDF\Legal\license directory.
- Review <National Instruments>\\_Legal Information.txt for information on including legal information in installers built with NI products.

## U.S. Government Restricted Rights

If you are an agency, department, or other entity of the United States Government ("Government"), the use, duplication, reproduction, release, modification, disclosure or transfer of the technical data included in this manual is governed by the Restricted Rights provisions under Federal Acquisition Regulation 52.227-14 for civilian agencies and Defense Federal Acquisition Regulation Supplement Section 252.227-7014 and 252.227-7015 for military agencies.

## Trademarks

Refer to the *NI Trademarks and Logo Guidelines* at [ni.com/trademarks](http://ni.com/trademarks) for more information on National Instruments trademarks.

ARM, Keil, and  $\mu$ Vision are trademarks or registered of ARM Ltd or its subsidiaries.

LEGO, the LEGO logo, WEDO, and MINDSTORMS are trademarks of the LEGO Group.

TETRIX by Pitsco is a trademark of Pitsco, Inc.

FIELDBUS FOUNDATION™ and FOUNDATION™ are trademarks of the Fieldbus Foundation.

EtherCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.

CANopen® is a registered Community Trademark of CAN in Automation e.V.

DeviceNet™ and EtherNet/IP™ are trademarks of ODVA.

Go!, SensorDAQ, and Vernier are registered trademarks of Vernier Software & Technology. Vernier Software & Technology and `vernier.com` are trademarks or trade dress.

Xilinx is the registered trademark of Xilinx, Inc.

Taprite and Trilobular are registered trademarks of Research Engineering & Manufacturing Inc.

FireWire® is the registered trademark of Apple Inc.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix®, Tek, and Tektronix, Enabling Technology are registered trademarks of Tektronix, Inc.

The Bluetooth® word mark is a registered trademark owned by the Bluetooth SIG, Inc.

The ExpressCard™ word mark and logos are owned by PCMCIA and any use of such marks by National Instruments is under license.

The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

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/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patent Notice* at `ni.com/patents`.

## Export Compliance Information

Refer to the *Export Compliance Information* at `ni.com/legal/export-compliance` for the National Instruments global trade compliance policy and how to obtain relevant HTS codes, ECCNs, and other import/export data.

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

YOU ARE ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY AND RELIABILITY OF THE PRODUCTS WHENEVER THE PRODUCTS ARE INCORPORATED IN YOUR SYSTEM OR APPLICATION, INCLUDING THE APPROPRIATE DESIGN, PROCESS, AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

PRODUCTS ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING IN THE OPERATION OF NUCLEAR FACILITIES; AIRCRAFT NAVIGATION; AIR TRAFFIC CONTROL SYSTEMS; LIFE SAVING OR LIFE SUSTAINING SYSTEMS OR SUCH OTHER MEDICAL DEVICES; OR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, PRUDENT STEPS MUST BE TAKEN TO PROTECT AGAINST FAILURES, INCLUDING PROVIDING BACK-UP AND SHUT-DOWN MECHANISMS. NI EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES.

# Contents

---

## About This Manual

Related Documentation .....	ix
-----------------------------	----

## Chapter 1

### Installing the Vision Builder AI Development Toolkit

Introduction .....	1-1
System Requirements .....	1-1
Installation Instructions .....	1-3
Installation Instructions If Vision Builder AI Is Already Installed .....	1-3
Importing Previous Vision Builder AI Custom Steps .....	1-4

## Chapter 2

### Understanding Custom Steps

Types of Custom Steps .....	2-1
Targets for Custom Steps .....	2-2
Custom Step Files .....	2-2
Source Code VIs .....	2-2
All VIs VI .....	2-2
Init Globals VI .....	2-3
Parameters Control .....	2-4
User Interface VI .....	2-4
User Programming VI .....	2-4
Utility VIs .....	2-5
VBAI Check Unique Step Name VI .....	2-5
VBAI CoordSys ID Utility VI .....	2-5
VBAI CoordSys Name Utility VI .....	2-5
VBAI Decision Maker (Float) VI .....	2-6
VBAI Display in Main Window VI .....	2-6
VBAI Get Global Variables 2 VI .....	2-6
VBAI SDK - Create Result VI .....	2-6
VBAI SDK Get All Images VI .....	2-6
VBAI SDK Get All Results VI .....	2-6
VBAI SDK Get Result 2 VI .....	2-6
VBAI SDK Modbus Register Manager .....	2-7
VBAI SDK - Populate Results Ring VI .....	2-7
VBAI SDK WindSetROI VI .....	2-7
Custom Step Concepts .....	2-7
Examples .....	2-8
Setup Variant .....	2-9
Process ID .....	2-9

## Chapter 3

### Creating a Custom Image Processing Step

Generating a Custom Step from a Template.....	3-1
Accessing the Custom Step Source VIs.....	3-3
Modifying the Custom Step Source VIs.....	3-3
Modifying the User Interface VI .....	3-3
Adding Threshold Range Controls .....	3-4
Initializing Values for the Threshold Range Controls .....	3-5
Modifying the User Programming VI.....	3-6
Preparing the Custom Step for Distribution .....	3-8
Changing the Tab Location of the Custom Step.....	3-8
Customizing the Custom Step Icon .....	3-8
Creating Documentation for the Custom Step.....	3-8
Debugging the Custom Step .....	3-8
Configuring LabVIEW to Debug Custom Steps .....	3-9
Debugging the Custom Step .....	3-9
Distributing The Custom Step .....	3-11

## Chapter 4

### Logging Measurement and Setting Limit Conditions

Generating the Custom Step from an Existing Step.....	4-1
Accessing the Custom Step Source VIs.....	4-3
Logging Measurements.....	4-3
Calculating the Threshold Percentage .....	4-3
Storing the Threshold Percentage in the Measurements Array .....	4-5
Returning Pass/Fail Data .....	4-6
Adding Pass/Fail Controls .....	4-7
Initializing Values for the Pass/Fail Controls.....	4-10
Initializing Default Values for the Pass/Fail Controls .....	4-10
Saving Values for the Pass/Fail Controls .....	4-11
Initializing Existing Values for the Pass/Fail Controls.....	4-12
Handling Value Changes for the Pass/Fail Controls .....	4-13
Setting the Step Status Based on Pass/Fail Analysis .....	4-14
Updating the User Interface with Measurement Data.....	4-15
Debugging the Custom Step .....	4-17

## Chapter 5

### Using Previous Measurements

Generating the Custom Step from an Existing Step .....	5-1
Accessing the Custom Step Source VIs .....	5-2
Modifying the Type Definition.....	5-2
Adding Previous Measurement Controls.....	5-4
Enumerating Previous Measurements .....	5-6
Initializing Values for Previous Measurement Controls .....	5-7
Initializing Default Values for Previous Measurement Controls .....	5-7
Initializing Existing Values for Previous Measurements Controls .....	5-8
Handling Value Changes for Previous Measurement Controls.....	5-10
Handling a New Image .....	5-13
Performing an Image Threshold with Previous Measurement .....	5-13
Debugging the Custom Step .....	5-16

## Appendix A

### Controls and Indicators Used in Source Code VIs

Init Globals VI Parameters .....	A-1
User Interface VI Parameters .....	A-3
User Programming VI Parameters.....	A-6

## Appendix B

### Creating Documentation for the Custom Step

Creating HTML Documentation for the Custom Step .....	B-1
Downloading and Installing HTML Workshop.....	B-1
Integrating the Custom Step Documentation.....	B-2

## Appendix C

### NI Services

# About This Manual

---

This manual contains the following information related to the Vision Builder for Automated Inspection (Vision Builder AI) Development Toolkit:

- System requirements and installation instructions
- Descriptions of the source code VIs for a custom step, and the interaction between source code VIs
- Tutorials that describe how to create and modify custom steps to process an image, perform pass/fail analysis, and use measurements from previous steps in the inspection



**Note** The Vision Builder AI Development Toolkit is designed for advanced LabVIEW users who have experience developing LabVIEW applications with the NI Vision Development Module.

## Related Documentation

---

The following documents contain information that you may find helpful as you read this manual:

- *NI Vision Concepts Help*—Describes the basic concepts of image analysis, image processing, and machine vision. This document also contains in-depth discussions about imaging algorithms for advanced users. The *NI Vision Concepts Help* is available by selecting **Start»All Programs»National Instruments»Vision Builder AI»Documentation»NI Vision Concepts Help** from the Start menu.
- *NI Vision Builder for Automated Inspection: Configuration Help*—Contains information about using the Vision Builder for Automated Inspection Configuration Interface to create a machine vision application. The *NI Vision Builder for Automated Inspection: Configuration Help* is available by selecting **Start»All Programs»National Instruments»Vision Builder AI»Documentation»Vision Builder AI Configuration Interface Help** from the Start menu.
- *NI Vision Builder for Automated Inspection: Inspection Help*—Contains information about running applications created with Vision Builder AI Inspection Interface. The *NI Vision Builder for Automated Inspection: Inspection Help* is available by selecting **Start»All Programs»National Instruments»Vision Builder AI»Documentation»Vision Builder AI Inspection Interface Help** from the Start menu.
- *NI Vision Builder for Automated Inspection Tutorial*—Describes Vision Builder for Automated Inspection and provides step-by-step instructions for solving common visual inspection tasks, such as inspection, gauging, part presence, guidance, and counting. The *NI Vision Builder for Automated Inspection Tutorial* is available by selecting **Start»All Programs»National Instruments»Vision Builder AI»Documentation»Vision Builder AI Tutorial** from the Start menu.
- *NI Vision for LabVIEW Help*—Contains reference information about NI Vision for LabVIEW palettes and VIs. This help file also guides you through tasks, from setting up your imaging system to taking measurements. If the NI Vision Development Module is



installed the *NI Vision for LabVIEW Help* is available by selecting **Help»NI Vision for LabVIEW Help** from the LabVIEW interface.

- **Support**—Technical support at [ni.com/support](http://ni.com/support) includes the following resources:
  - **Self-Help Resources**—For answers and solutions, visit [ni.com/support](http://ni.com/support) for software drivers and updates, a searchable KnowledgeBase at [ni.com/kb](http://ni.com/kb), product manuals at [ni.com/manuals](http://ni.com/manuals), step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on. Registered users also receive access to the NI Discussion Forums at [ni.com/forums](http://ni.com/forums). NI Applications Engineers make sure every question submitted online receives an answer.
  - **Standard Service Program Membership**—This program entitles members to direct access to NI Applications Engineers via phone and email for one-to-one technical support, as well as exclusive access to self-paced online training modules at [ni.com/self-paced-training](http://ni.com/self-paced-training). All customers automatically receive a one-year membership in the Standard Service Program (SSP) with the purchase of most software products and bundles including NI Developer Suite. NI also offers flexible extended contract options that guarantee your SSP benefits are available without interruption for as long as you need them. Visit [ni.com/ssp](http://ni.com/ssp) for more information.
- **Training and Certification**—Visit [ni.com/training](http://ni.com/training) for training and certification program information. You can also 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).

You can also visit the Worldwide Offices section of [ni.com/niglobal](http://ni.com/niglobal) to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

---

# Installing the Vision Builder AI Development Toolkit

This chapter introduces the Vision Builder for Automated Inspection (Vision Builder AI) Development Toolkit. This chapter also contains the system requirements and installation instructions for the Vision Builder AI Development Toolkit.

## Introduction

---

The Vision Builder AI Development Toolkit features a wizard and an application program interface (API) to create custom steps for use in any Vision Builder AI inspection.



**Note** This toolkit is designed for advanced LabVIEW users who have experience developing LabVIEW applications with the NI Vision Development Module.

A custom step processes an image according to parameters or limits, which may be defined by the user, and produces results. You can create custom steps to provide functionality that is not already included in Vision Builder AI. For example, you can develop a custom step that drives a camera or other hardware not currently supported in Vision Builder AI. You also can create a custom step to include a customized image processing algorithm.



**Note** The Vision Builder AI Development Toolkit does not support the creation of acquisition steps for remote targets.

You can distribute custom steps with the executable version of Vision Builder AI, to provide end users with a customized Vision Builder AI package.

## System Requirements

---

Table 1-1 includes minimum system requirements for the Vision Builder AI Development Toolkit.

**Table 1-1.** Minimum System Requirements

	Minimum	Recommended
<b>Processor</b>	Pentium 4/M or equivalent	Core Duo or equivalent
<b>Memory</b>	512 MB RAM	1 GB RAM

**Table 1-1. Minimum System Requirements (Continued)**

	<b>Minimum</b>	<b>Recommended</b>
<b>Display</b>	1,024 × 768 resolution video adapter with a 24-bit or 32-bit display	1,280 × 1,024 resolution video adapter with a 24-bit or 32-bit display
<b>Operating System</b>	<ul style="list-style-type: none"> <li>• Windows 10 (32- and 64-bit)</li> <li>• Windows 10 IoT Enterprise (64-bit) <sup>a</sup></li> <li>• Windows 8.1 (32- and 64-bit) <sup>b</sup></li> <li>• Windows 7 SP1 (32- and 64-bit) <sup>c</sup></li> <li>• Windows Embedded Standard 7 SP1 <sup>a, c</sup></li> <li>• Windows Server 2012 R2 (64-bit) <sup>b</sup></li> <li>• Windows Server 2008 R2 SP1 (64-bit) <sup>c</sup></li> </ul>	
<b>Browser</b>	Microsoft Internet Explorer 6.0 or later	
<b>Free Hard Disk Space</b>	1 GB	
<b>Software</b>	LabVIEW 2018 NI Vision Development Module 2018 <sup>d</sup> NI Vision Acquisition Software 2018 <sup>e</sup> LabVIEW Real-Time 2018 or later <sup>f</sup> Vision Builder AI 2018	

a. Only supported as pre-installed on NI hardware.

b. NI software installs VC2015 Runtime and .NET 4.6.2. Windows 8.1 and Windows Server 2012 R2 require Microsoft updates to support these items. Refer to Microsoft KB2919442 and KB2919355 for more information about how to install these updates.

c. NI software is signed with a SHA-256 certificate. Windows 7 SP1, Windows Embedded Standard 7 SP1, and Windows Server 2008 R2 SP1 require Microsoft updates to support SHA-256. Refer to Microsoft KB3033929 for more information about how to install this security update.

d. The NI Vision Development Module is required to develop custom image processing steps.

e. The NI Vision Acquisition Software is required to deploy custom image processing steps to remote targets.

f. The LabVIEW Real-Time Module is required to develop custom steps for remote targets. The Vision Builder AI Development Toolkit does not support the creation of acquisition steps for remote targets.

# Installation Instructions

---

Complete the following steps to install the Vision Builder AI Development Toolkit.



**Note** You must install LabVIEW before installing the Vision Builder AI Development Toolkit.



**Note** To install the Vision Builder AI Development Toolkit on a Windows system, you must be logged in with administrator privileges.

1. In the feature tree for Vision Builder AI, select **Development Toolkit** and select **Install this feature to a local drive**.
2. Click **Next**.
3. Follow the setup instructions on your screen.

The Vision Builder AI Development Toolkit files are installed in the <LabVIEW>\project\Vision Builder AI directory, where <LabVIEW> is the location to which LabVIEW is installed.

## Installation Instructions If Vision Builder AI Is Already Installed

Complete the following steps to install the Vision Builder AI Development Toolkit on a system that already has Vision Builder AI installed.



**Note** You must install LabVIEW before installing the Vision Builder AI Development Toolkit.

1. Locate the **National Instruments Software** installation.
  - **Windows XP/Server 2003 R2 (32-bit)**—Select **Start»Control Panel»Add or Remove Programs**
  - **Windows 7/Vista/Server 2008 R2**—Select **Start»Control Panel»Programs»Uninstall a Program**
  - **Windows 8**—Right-click the Start screen, select **All apps**, and launch the Control Panel. Select **Programs and Features»Uninstall/Change**
2. Select the **National Instruments Software** installation, then click **Change**.
3. Select **NI Vision Builder AI** and click **Modify**.
4. In the feature tree for Vision Builder AI, select **Development Toolkit** and select **Install this feature to the local drive**.
5. Click **Next**.
6. Follow the setup instructions on your screen.

The Vision Builder AI Development Toolkit files are installed in the `<LabVIEW>\project\Vision Builder AI` directory, where `<LabVIEW>` is the location to which LabVIEW is installed.

## Importing Previous Vision Builder AI Custom Steps

---

Complete the following steps to import steps created for Vision Builder AI 2013.

1. Launch LabVIEW.
2. Close any open VIs.
3. Select **Tools»<Vision Builder AI>»Import Vision Builder AI Step**.
4. Custom steps found in the `<Vision Builder AI previous version>\UserPlugins` folder are listed in the **Existing Custom Steps Previously Created** textbox.
5. Select the step you want to import.
6. Click **Next**.
7. Click **Finish**.

The Import Vision Builder AI Step wizard creates the following files in the `<Vision Builder AI>\UserPlugins` folder:

- `59 <Custom Step Name>.llb`—Custom step source code
- `<Custom Step Name>.tif`—Custom step icon

---

# Understanding Custom Steps

This chapter describes the components of a custom step created with the Vision Builder AI Development Toolkit.

## Types of Custom Steps

---

The Vision Builder AI Development Toolkit includes several templates that you can modify to create a custom step. When you create a new custom step, you can select from the following templates:

- **Simulated Acquisition Step**—Acquires or creates an image and makes it available to other steps for processing.  
You must start with the Simulated Acquisition Step template when you create custom acquisition steps.
- **Simple Processing Step**—Processes the image from the previous step and returns only pass/fail information.
- **Processing Step that Logs Measurements**—Processes an image from the previous step, makes measurements in the image, and makes the measurements available to subsequent steps.
- **Generate a Report**—Provides access to the measurements produced by previous steps. You can use previous measurements to create new measurements and create a customized pass/fail condition.
- **Global Step Status**—This step passes inspection if all of the previous steps passed. Otherwise, the step fails inspection. This step does not require the NI Vision Development Module. If you are developing a custom step, and you do not have the NI Vision Development Module installed, select this template. This step demonstrates how to use the Vision Builder AI Development Toolkit without the NI Vision Development Module.
- **Coordinate System**—This step uses coordinate systems to reposition regions of interest.



**Tip** Examine other templates to help determine how to modify your custom step. For example, if you want to create a custom step that combines image processing and creating image masks, use the Simple Processing template to create the custom step, then refer to the Coordinate System template for ideas about how to implement image mask creation.

# Targets for Custom Steps

---

Custom steps can be deployed to various targets for use in an inspection. The Vision Builder AI Development Toolkit will build your custom step for distribution to the following targets:

- **Windows PC**—A Windows PC with an executable version of Vision Builder AI can use your custom step as part of an inspection.
- **Embedded Vision System**—An automated controller that acquires images from an array of cameras and performs an inspection.
- **Smart Camera (PPC)**—An NI Smart Camera (excluding NI 177x Smart Cameras) that acquires images, performs an inspection, and transmits the results.
- **Smart Camera (Atom)**—An NI 177x Smart Camera that acquires images, performs an inspection, and transmits the results.
- **Compact Vision System**—A real-time compact vision system that acquires, processes, and displays images from cameras. NI Compact Vision Systems run on LabVIEW Real-Time or NI Linux Real-Time.

## Custom Step Files

---

When you generate a new custom step, the Create Custom Step wizard creates two files in the <Vision Builder AI>\UserPlugins folder:

- 59 <Custom Step Name>.llb—Contains the source code for the custom step, as well as several utility VIs that you can use as you modify your custom step. If you create helper VIs or controls for your custom step, save them in this LLB. Refer to the [Source Code VIs](#) section for more information about the source code VIs stored in the custom step LLB.
- <Custom Step Name>.tif—Icon for the custom step. Modify this file to change the icon for the custom step. Do not change the file extension or dimensions of the icon.

To deploy a custom step, you must save the custom step for distribution. Refer to the [Distributing The Custom Step](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for more information about deploying custom steps.

## Source Code VIs

This section describes the source code VIs stored within the custom step LLB. Modify the source code VIs to customize the behavior of your custom step.

### All VIs VI

The All VIs VI is not designed to be executed. The block diagram of the All VIs VI provides a project window that contains the VIs that you can modify or use in your custom step.

The block diagram of this VI is divided into the following sections:

- **Set Up Global**—Contains the <Custom Step Name> - Init Globals VI. Refer to the [Init Globals VI](#) section for more information about this VI.

The Init Globals VI is connected to a string indicator to ensure that the All VIs VI has no errors. This string indicator serves no other purpose.

- **Custom VIs**—This section contains the VIs you must modify to customize the behavior of your custom step. Refer to the [User Interface VI](#) section and the [User Programming VI](#) section for more information about custom VIs.
- **Utility VIs**—This section contains utility VIs that you can call within the source VIs to perform specific functions, such as displaying an image in the main Vision Builder AI window. You cannot modify these VIs. Refer to the [Utility VIs](#) section for more information about utility VIs.
- **Dynamic VIs**—This section is designed to contain any VIs that you call dynamically. Place any VIs that you call dynamically within a source VI in this section to ensure that the VI is saved correctly when you build the custom step for distribution.



**Note** The VIs in a custom step use the custom step name as a prefix. To create a reference to a dynamic VI in a Run Time System, use the custom step name followed by an underscore as a prefix, such as `Custom Step_Dynamic VI.vi`



**Tip** To distinguish between development mode and Run Time System, use the `App.Kind` property node to open a reference to a dynamic VI.



**Note** If you are building a custom step for real-time targets, you must add all dynamic VIs to the `<Step Name> - All RT VIs.vi`

## Init Globals VI

The Init Globals VI allows you to specify the following properties for a custom step:

- The name and description of the step that appear in Vision Builder AI.
- The version of the custom step.
- The tools palette tab that contains the custom step in Vision Builder AI.
- The region where the user interface for the custom step appears in Vision Builder AI.
- Whether the custom step can be used in product selection mode.
- The path to a help file that you create for the custom step. Refer to Appendix B, [Creating Documentation for the Custom Step](#), for information about creating help for the custom step.
- The default ROI tool for the step, the ROI tools supported by the custom step, the names of the available ROI tools, and the display of previously created ROIs.
- The image file types supported by the custom step.
- If the step produces an image, you can specify for the step to log the modified image and make it available in the **Select Image** step.



For detailed information about the parameters in the Init Globals VI that you can set to specify the properties of your custom step, refer to Appendix A, *Controls and Indicators Used in Source Code VIs*.



**Note** When you modify the controls and indicators in the Init Globals VI, you must make the new values the default for the control or indicator and save the changes. To make the current values the default for a control or indicator, select **Edit**►**Make Current Values Default**, and save the VI.

## Parameters Control

This is the type definition for the parameters used by the custom step. Type definitions link all the instances of a custom control or indicator to a saved custom control or indicator file. You can update all instances of a parameter by editing the <Custom Step Name> - parameters.ct1 file, which allows you to easily use the same controls and indicators in multiple source code VIs.



**Tip** If you have multiple parameters, place them inside a cluster. By bundling several data elements into a single cluster data structure, you eliminate wire clutter on the block diagram and reduce the number of VI connector pane terminals required to handle parameter data.

## User Interface VI

The front panel of the User Interface VI is the interface the user sees when configuring the custom step. The User Interface VI allows the user to modify step parameters, and can compare data from the User Programming VI with user-defined parameters to indicate the step status.

The User Interface VI is executed when a user selects the custom step from the Vision Builder AI Inspection Steps palette or when the user opens a step already inserted in the script to edit parameters.



**Note** Do not resize the tab control on the user interface or change its position. Changing the tab control on the custom step user interface might move the user interface controls and indicators to a position that the user cannot access.



**Tip** To display the user interface of the custom step in the main window of Vision Builder AI, open the Init Globals VI and select **Main Window** from the **Region** list.

## User Programming VI

The User Programming VI performs image processing, and can return image processing results. The front panel is never displayed, so it does not have to be modified. Do *not* modify the connector pane of this VI.

The Vision Builder AI engine sets the value of the Setup variant control, which contains the setup parameters defined by the user in the user interface. Use the Variant To Data VI to access

this data. Use the same data structure that you used to package the data in User Interface VI. To ensure that you use the same data type, always use the Parameters control type definition with the Variant to Data VI.

The User Programming VI operates in one of the three following modes:

- **Setup**—Executed once when the user opens an inspection that contains the custom step or inserts the custom step in the script.  
Use this mode to perform any one-time initialization routines, such as reading a pattern matching template or initializing a data acquisition (DAQ) or image acquisition device.
- **Execution**—Executed each time the step is called within the Vision Builder AI inspection.  
This mode performs the main function of the custom step.
- **Cleanup**—Executed when the user closes an inspection or when the step is removed from the script.  
Use this mode to dispose of any resources allocated in the Setup mode.

## Utility VIs

---

Utility VIs perform tasks that are commonly necessary in Vision Builder AI steps. These VIs can be accessed from the All VIs block diagram, but not all utility VIs are included with every type of custom step. The utility VIs are located at <Vision Builder AI>\Plugins\Common SDK VIs.llb.

### VBAI Check Unique Step Name VI

Use this VI to verify that a step with the same name has not previously been inserted into the inspection. Vision Builder AI does not allow duplicate step names. To see how this VI is used, open the User Interface VI and examine the "OK": Value Change case of the event structure.

### VBAI CoordSys ID Utility VI

Use this VI in a custom Coordinate System step to get the measurement system data needed to reposition the region of interest.

The input to the VBAI CoordSys ID Utility VI is a unique coordinate system ID. The unique coordinate system ID ensures that the correct coordinate system is selected, even when the coordinate system is renamed. The VBAI CoordSys ID Utility VI returns the name of the coordinate system as it appears in the user interface, along with the coordinate system data. The VBAI CoordSys ID Utility VI is also used to update the user interface from the saved coordinate system ID.

### VBAI CoordSys Name Utility VI

Use this VI in a custom Coordinate System step to get the coordinate system identifier and measurement system that you need to save in the step parameters.

The input to the VBAI CoordSys Name Utility VI is a coordinate system identified by the coordinate system name that the user selected from the user interface. The VBAI CoordSys Name Utility VI returns a unique ID for the coordinate system, along with the coordinate system data. The unique coordinate system ID ensures that the correct coordinate system is selected, even when the coordinate system is renamed.

## VBAI Decision Maker (Float) VI

This VI returns pass/fail information by applying the pass/fail conditions specified in the custom step user interface to the pass/fail conditions data of the User Programming VI. Use the VBAI Decision Maker VI only in the User Interface VI so that the user can see immediate pass/fail results when modifying the pass/fail conditions. Outside of the user interface, the decision making functionality is performed automatically.

## VBAI Display in Main Window VI

This VI is only used in the User Interface VI. Use this VI in the User Interface VI to update the Main window of Vision Builder AI and interactively show changes in the image after parameter values are changed.

## VBAI Get Global Variables 2 VI

This VI returns the currently defined variable and custom Inspection Interface control values.

## VBAI SDK - Create Result VI

Use this VI in the User Programming VI to properly bundle measurements and pass/fail result strings as clusters that can be easily inserted into the Measurements and Pass/Fail Data arrays.

The VBAI SDK - Result VI is a polymorphic VI that will automatically switch to the correct instance when you connect a measurement to the Result Value input. Valid Result Value input types are Boolean, numeric, or string values; 1D Boolean arrays; 1D numeric arrays; or 1D string arrays.

## VBAI SDK Get All Images VI

Use this VI to get all images produced by Vision Builder AI steps.

## VBAI SDK Get All Results VI

Use this VI to get a 1D array that contains previous measurement data from all states, as well as any variables declared in Vision Builder AI.

## VBAI SDK Get Result 2 VI

Use this VI to get result information from previous steps. This VI returns result information, such as the name and value of the result, that corresponds to a Measurement ID.

## VBAI SDK Modbus Register Manager

Use this VI to access the Vision Builder AI Modbus registers from your custom step.

## VBAI SDK - Populate Results Ring VI

Use this VI to populate a ring control with a list of previous results. This VI loops through the results returned by previous steps in the inspection and returns a list of results of the selected data type. The default data type is numeric.

## VBAI SDK WindSetROI VI

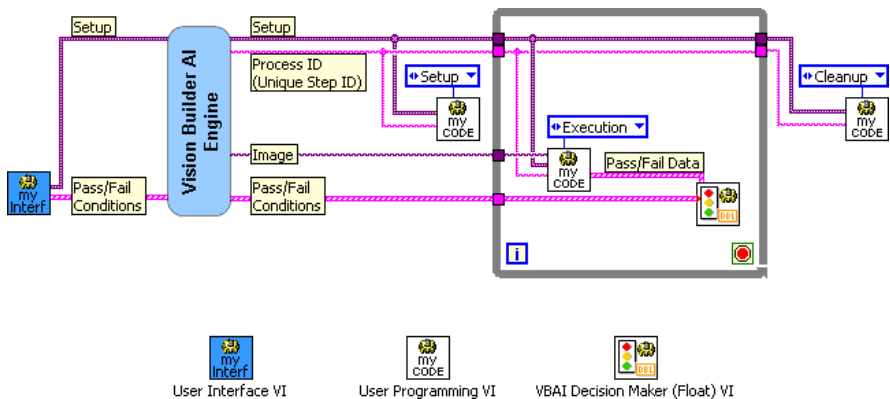
Use this VI in the User Interface VI to set an ROI in the image, based on an ROI descriptor.

## Custom Step Concepts

A custom step depends primarily on two source code VIs: the User Interface VI and the User Programming VI. You must understand how these source code VIs exchange data so that you can efficiently modify the source code VIs and customize the behavior of your custom step.

Figure 2-1 shows how the custom step configuration data flows from the user interface to the execution of the custom step, which occurs in a loop. The outputs of the User Interface VI are saved in the Vision Builder AI engine, which then dispatches the outputs to the inputs of the User Programming VI.

**Figure 2-1.** Dataflow between the User Interface, User Programming, and Utility VIs



The User Interface VI is called when the user selects the custom step from a palette, or opens the custom step to edit its configuration. The front panel of the User Interface VI is the interface that the user sees when configuring the custom step.

If the user validates a custom step by clicking OK in the user interface of the custom step, Vision Builder AI automatically calls the setup and execution modes of the User Programming VI and inserts the step in the script.



**Note** When developing the User Interface VI, ensure that you call the User Programming VI in setup mode to perform any initialization before you call the execution mode of the User Programming VI.

When the step executes, the Vision Builder AI engine sets the image and the Previous Measurements array, and sets the Setup variant to the value previously set in User Interface VI. In execution mode, the User Programming VI accesses the data from the User Interface VI, then processes the image and produces results.

When the user closes the script, the Vision Builder AI engine calls the cleanup mode of the User Programming VI. Figure 2-1 illustrates the sequence of the setup, execution, and cleanup modes of the User Programming VI.

## Examples

Complete the following steps to view a block diagram that shows the event structure that handles the **OK** button in the user interface of the custom step:

1. Create a custom step using the Simple Processing Step template.
2. Open the User Interface VI.
3. View the block diagram.
4. Select frame 3 of the sequence structure.
5. Select event case 3 of the event structure. In this event case, the step uses the VBAI Check Unique Step Name utility VI to verify that the step name is valid before it allows the step to exit.

Complete the following steps to view a block diagram that shows the relationship between the User Interface VI and the User Programming VI:

1. Create a custom step using the Processing Step that Logs Measurements template.
2. Open the User Interface VI.
3. View the block diagram.
4. Select frame 3 of the sequence structure.
5. Select event case 0 of the event structure. In this event case, the step compares the Pass/Fail Data returned from the User Programming VI to the Pass/Fail Conditions set by the user in the user interface, then updates the Step Status indicator based on the comparison results. The step also uses the VBAI Display in Main Window utility VI to update the main window of Vision Builder AI with the processed image returned by the User Programming VI.

## Setup Variant

The Setup variant stores the parameters of the step. If the step requires more than one parameter, bundle the parameters in a cluster, and use the To Variant VI to store the parameters in the variant. Use the Variant to Data VI to retrieve these parameters in the User Programming VI.

Complete the following steps to view a block diagram that shows the relationship between the step parameters and the Variant to Data VI:

1. Create a custom step using the Simple Processing Step template.
2. Open the User Interface VI.
3. View the block diagram.
4. Select frame 1 of the sequence structure.
5. Select the True case of the case structure. The True case executes if the user edits the step, and updates the user interface with values stored in the Setup in variant.

## Process ID

When the user creates an instance of a custom step, the Vision Builder AI engine provides the step instance with a Process ID, which is a string that uniquely identifies the custom step. The Process ID differentiates multiple instances of the same step.

Use the Process ID to allocate resources to a specific instance of a custom step. For example, you can pass a Process ID to the IMAQ Create VI to allocate a unique image for an instance of a custom step. The IMAQ Create VI is installed as part of the NI Vision Development Module or NI Vision Acquisition Software.



**Tip** The VBAI Resource Manager utility VI is designed to help you create, get, and deallocate step specific resources. Refer to the [Utility VIs](#) section for more information about the VBAI Resource Manager VI.



**Tip** You can use a variable to pass information between the different modes of the VI. If the values of the variable change between multiple instances of the same custom VI, the variable is overwritten with the latest value.

Save any global variables created in the 59 <Custom Step Name>.llb with the step name as the prefix, followed by a dash. For example, if you created a variable called MyGlobal for a step you named Image Processing, save the global variable as `Image Processing - MyGlobal.vi`.

---

# Creating a Custom Image Processing Step

This chapter introduces concepts required to create custom steps for Vision Builder AI.



**Note** The Vision Builder AI Development Toolkit does not support the creation of acquisition steps for remote targets, such as the NI 17xx Smart Camera or NI EVS-1460 Series Embedded Vision System.

Follow the instructions in this chapter to complete the following tasks:

- Generate a custom step from a template using the Create Custom Step Wizard
- Modify the custom step to perform an image threshold
- Debug the custom step
- Save the custom step for deployment



**Note** This tutorial requires the NI Vision Development Module in addition to the minimum system requirements of the Vision Builder AI Development Toolkit.

---

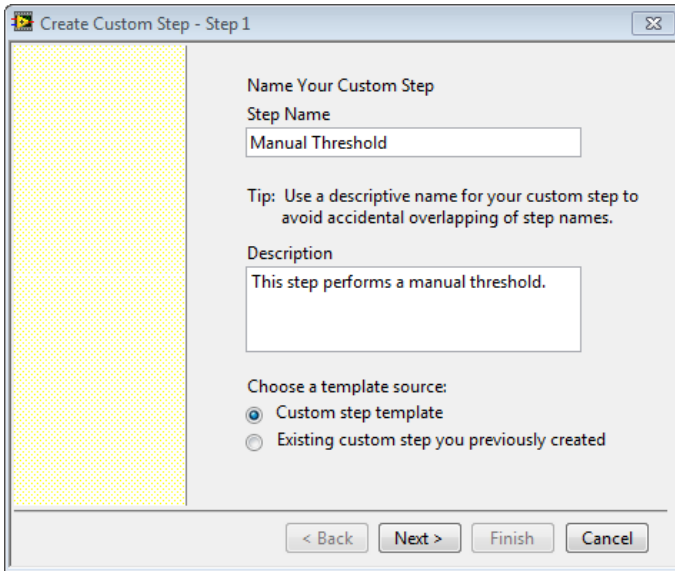
## Generating a Custom Step from a Template

Complete the following steps to generate a custom step using the Create Custom Step wizard:

1. Launch LabVIEW.
2. Select **Tools»<Vision Builder AI>>Create Custom Step**.

3. Enter `Manual Threshold` as the **Step Name** for the custom step. Figure 3-1 illustrates the Step Name text box.

**Figure 3-1.** Naming a Custom Step



4. Enter `This step performs a manual threshold` as the **Description** for the custom step. The description is displayed next to the step name in the Vision Builder AI Inspection Steps palette.
5. Click **Next**.
6. In the **Templates** list select **Simple Processing Step**. A simple processing step processes an input image and returns pass/fail information. Refer to the *Types of Custom Steps* section of Chapter 2, *Understanding Custom Steps*, for information about each custom step template.
7. Click **Next** and verify the setup information. Close all open VIs to ensure that the custom step is created successfully.
8. Click **Finish** to create the custom step.
9. The Create Custom Step wizard opens the Manual Threshold - All VIs VI block diagram. This VI provides a project window that contains the VIs that you can modify for the custom step. Refer to the *Source Code VIs* section of Chapter 2, *Understanding Custom Steps*, for information about the All VIs VI.



**Note** The custom step is not available in Vision Builder AI until you save it for distribution. National Instruments recommends that you debug the custom step before you save it for distribution. Refer to the *Debugging the Custom Step* section for information about debugging custom steps.



## Accessing the Custom Step Source VIs

---

The Create Custom Step wizard generates source VIs that provide the functionality of the custom step. The Create Custom Step wizard installs the source VIs to the `<Vision Builder AI>\UserPlugins\59 Manual Threshold.llb` file. Use the All VIs VI block diagram to access the custom step source VIs.

Complete the following steps to open the All VIs VI for the custom step, if it is not already open:

1. Launch LabVIEW.
2. Select **File»Open**.
3. Navigate to `<Vision Builder AI>\UserPlugins\59 Manual Threshold.llb` file.
4. Open the LLB.
5. Select the `Manual Threshold - All VIs.vi` and click **OK**.

## Modifying the Custom Step Source VIs

---

You must modify the following source VIs to specify custom behavior for the custom step:

- `Manual Threshold - User Interface.vi`—This VI contains the user interface of the custom step.



- `Manual Threshold - User Programming.vi`—This VI performs the initialization, task, and cleanup phases of the custom step.



## Modifying the User Interface VI

The User Interface VI is called when a user selects the custom step from the Vision Builder AI Inspection Steps palette or when the user opens a step already inserted in the script to edit parameters.

Follow the instructions in this section to complete the following tasks:

- Add controls to the User Interface VI that allow the user to specify the threshold range
- Modify the block diagram of the User Interface VI to set default values for the threshold range



**Note** Do not resize the tab control on the user interface or change its position. Changing the tab control on the custom step user interface might move the user interface controls and indicators to a position that the user cannot access.

## Adding Threshold Range Controls

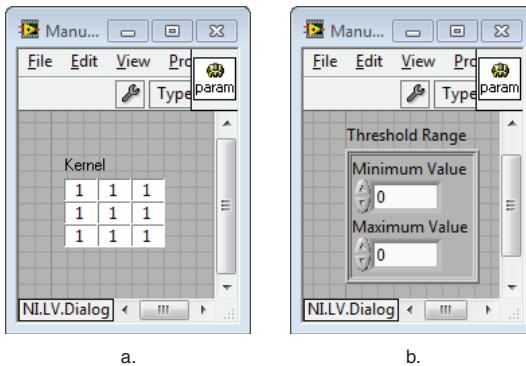
Complete the following steps to add controls to the User Interface VI that allow the user to specify the threshold range:

1. Open the All VIs VI.
2. Double-click the User Interface VI to open it.



3. Select the **Settings** tab.
4. Locate the convolution kernel. The convolution kernel is the default control defined in the Parameters control, which is the type definition for the parameters used by the custom step. When you make changes in the type definition, every instance of the parameter is updated in the custom step code.
5. Right-click the convolution kernel and select **Open Type Def** to open the type definition.
6. Complete the following steps to edit the type definition to specify the threshold range parameters:
  - a. Select the convolution kernel and delete it. Figure 3-2a illustrates the default type definition.

**Figure 3-2.** Modifying the Type Definition

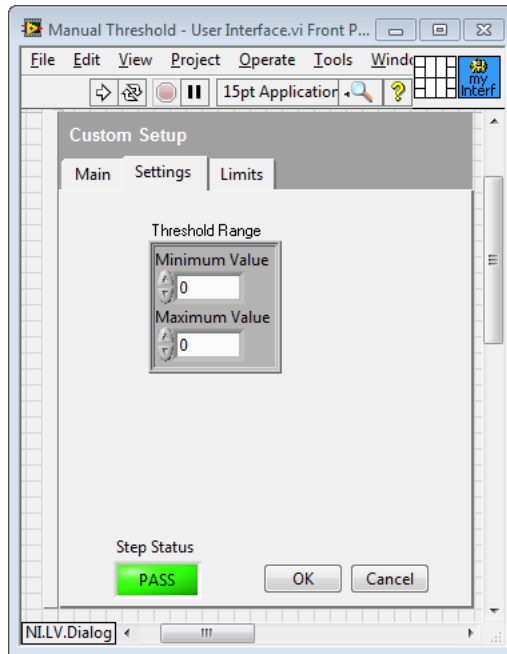


- b. Add a cluster to the front panel. Label the cluster `Threshold Range`.
  - c. Add a numeric control to the cluster. Label the numeric control `Minimum Value`.
  - d. Add another numeric control to the cluster. Label the numeric control `Maximum Value`, as illustrated in Figure 3-2b.
  - e. Select **File»Save** to save the type definition.
  - f. Select **File»Close** to close the type definition.
7. Position the type definition cluster so that the **Minimum Value** and **Maximum Value** controls are visible, as illustrated in Figure 3-3.



**Tip** Set the color of the cluster background and border to transparent to hide the cluster on the user interface. To change the cluster color to transparent, select **View»Tools Palette**, and select the **Set Color** tool. Right-click the cluster background or border to open the color picker, and Select the **T** box in the upper right corner of the color picker.

**Figure 3-3.** Editing the Settings Tab



8. Relabel the Kernel cluster as `Threshold Range`.
9. Select **File»Save** to save the User Interface VI.

The user interface now contains two numeric controls that allow the user to specify the range for the image threshold.

## Initializing Values for the Threshold Range Controls

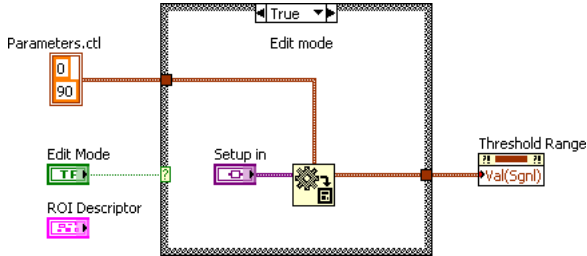
Complete the following steps to edit the block diagram of the User Interface VI and initialize the threshold range controls with default values.

1. With the User Interface VI front panel open, press <Ctrl-E> or select **Window»Show Block Diagram** to open the block diagram.
2. Select frame 1 of the Stacked Sequence structure.

Frames 0 and 1 initialize the custom step when it is loaded in Vision Builder AI. Frame 0 sets the default appearance of the user interface, and frame 1 initializes the user interface with default values.

3. Locate the Parameters.ctf cluster. The Parameters.ctf cluster contains two numeric controls that correspond to the threshold parameters defined in the type definition.
4. Enter 90 as the value for the **Maximum Value** numeric control. Figure 3-4 illustrates the modified control.

**Figure 3-4.** Editing Default Values for the Threshold Range



5. Select **File»Save** to save the User Interface VI.
6. Select **File»Close** to close the User Interface VI.

When the user adds the custom step to an inspection in Vision Builder AI, the User Interface VI initializes the Maximum Value control to 90.

## Modifying the User Programming VI

The User Programming VI contains the main processing code of the step. The front panel is never displayed, so it does not have to be modified.

Do *not* modify the connector pane of the User Programming VI.

Complete the following steps to modify the User Programming VI to perform an image threshold and return an updated image:

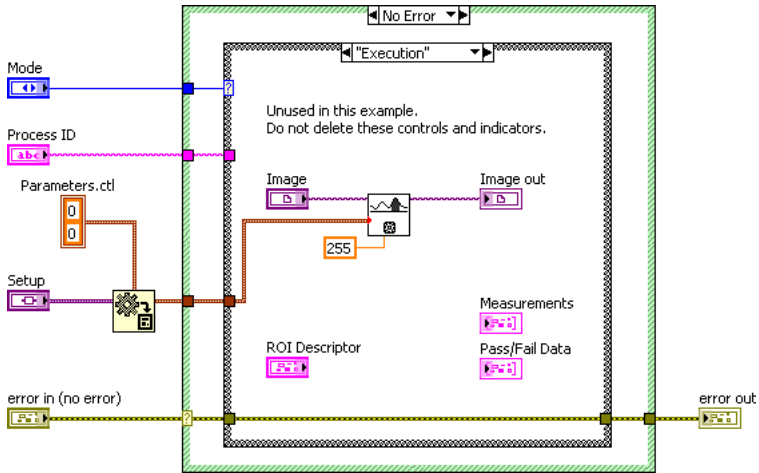
1. Open the All VIs VI.
2. Double-click the User Programming VI to open it.



3. Press <Ctrl-E> or select **Window»Show Block Diagram** and examine the block diagram.
4. Navigate to the No Error case of the case structure.
5. Navigate to the Execution case of the case structure within the No Error case. The Execution case provides the functionality of the step, and can process parameters or other data defined in the User Interface VI. Data from the user interface is passed in through the Setup variant.

6. Complete the following steps to build the block diagram shown in Figure 3-5:

**Figure 3-5.** Adding the Image Threshold Code



- a. The `Execution` case of the case structure contains a broken wire that connects the Variant to Data VI and the IMAQ Convolute VI. Delete the broken wire.
  - b. Replace the IMAQ Convolute VI with the IMAQ Threshold VI.
  - c. Verify that the **Image** control is connected to the **Image Src** input of the IMAQ Threshold VI.
  - d. Verify that **Image Dst Out** output of the IMAQ Threshold VI is connected to the **Image out** indicator.
  - e. Connect the **data** output from the Variant to Data VI to the **Range** input of the IMAQ Threshold VI.
  - f. Create a numeric constant with a value of 255 and connect it to the **Replace Value** input of the IMAQ Threshold VI.
7. Select **File»Save** to save the User Programming VI.
  8. Select **File»Close** to close the User Programming VI.

The Manual Threshold step is complete.

## Preparing the Custom Step for Distribution

---

You can customize the appearance of the custom step in Vision Builder AI and create documentation for the custom step.

### Changing the Tab Location of the Custom Step

Complete the following steps to change the tab location of the custom step in Vision Builder AI:

1. Open the All VIs VI.
2. Right-click the `Manual Threshold - Init Globals.vi` and select **Open Front Panel**.



3. In the **Tab Location** list, select the tab location for the step.
4. Select **Edit»Make Current Values Default**, and save the VI to make the current values the default for a control or indicator.



**Note** When you modify the controls and indicators in the Init Globals VI, you must make the new values the default for the control or indicator and save the changes.

5. Select **File»Save** to save the Init Globals VI.

### Customizing the Custom Step Icon

To customize the custom step icon, use a graphics editor to edit the `.tif` file associated with the custom step. The size of the custom step icon must be  $32 \times 32$  pixels. The custom step icon must be saved with the `.tif` file extension.

### Creating Documentation for the Custom Step

Refer to Appendix B, *Creating Documentation for the Custom Step*, for information about creating documentation for the custom step.

### Debugging the Custom Step

---

Follow the steps in this section to configure LabVIEW to debug custom steps and debug the Manual Threshold step.

## Configuring LabVIEW to Debug Custom Steps

To debug custom steps, you must enable TCP/IP support for the LabVIEW VI Server. Complete the following steps to enable TCP/IP support.

1. In LabVIEW, select **Tools»Options**.
2. Select **VI Server** from the Category list.
3. Enable the **TCP/IP** checkbox.
4. Enable all of the **Accessible Server Resources** options.
5. Verify that the **Machine access list** includes the IP address of the local host, or use \* to allow any machine to use the VI Server.
6. Verify that the **Exported VIs** list includes \*Proxy(Target) .vi, or use \* to allow all VIs to be exported.
7. Click **OK**.

LabVIEW is configured to debug custom steps.

## Debugging the Custom Step

Complete the following steps to open Vision Builder AI from LabVIEW and add the custom step to the inspection:

1. Launch LabVIEW.
2. Select **Tools»<Vision Builder AI>»Test Your Custom Step in Vision Builder AI** to launch Vision Builder AI and test the step.



**Note** You must open Vision Builder AI from LabVIEW to test a custom step that has not been saved for distribution. Refer to the [Distributing The Custom Step](#) section for information about saving custom steps for use with the Vision Builder AI executable.

3. Click **New Inspection** to open the Vision Builder AI Configuration interface.
4. Add a **Simulate Acquisition** step to the inspection. Select <Vision Builder AI>\DemoImg\tutorial 1\Image 01.jpg as the source image.  
Because the Manual Threshold custom step operates on an image, you must acquire an image to test the custom step.
5. Add the **Manual Threshold** custom step to the inspection. By default, custom steps are located on the **Use Additional Tools** palette.

If you did not modify the custom step icon, the custom step displays the default custom step icon.

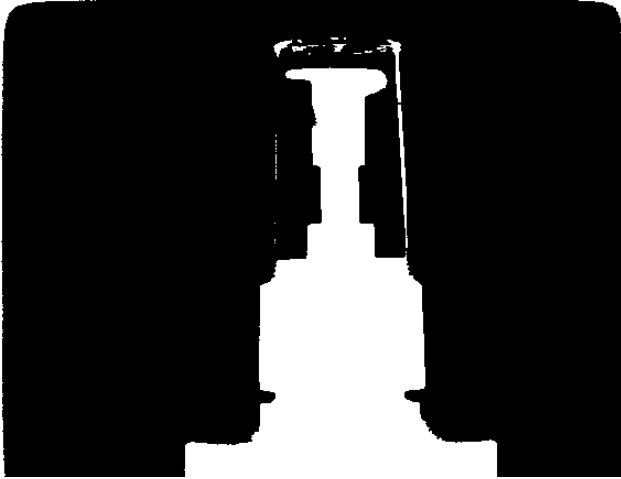


- When you open the custom step, Vision Builder AI operates in debugging mode. The front panel of the User Interface VI for the custom step opens in LabVIEW. The front panel of the User Interface VI may open behind the Vision Builder AI window. Open the User Interface VI by clicking the VI window in the taskbar.

When the custom step is added to the inspection, the main window displays the output image from the Manual Threshold custom step. Figure 3-6 illustrates the output image from the custom step. White pixels indicate values within the threshold range.

**Figure 3-6.** Output Image from the Custom Step

---



- Modify the **Minimum Value** and **Maximum Value**, and observe the changes made to the image in the main window of Vision Builder AI.
- Press <Ctrl-E> or select **Window»Show Block Diagram** to open the block diagram and begin debugging the custom step.  
Standard LabVIEW debugging tools are available to debug the custom step. Select the appropriate debugging options and breakpoints to debug the custom step.
- Exit debugging mode by clicking **Abort Execution** in the LabVIEW toolbar, by clicking **Cancel** in the custom step User Interface VI front panel, or by clicking the **Exit** button in Vision Builder AI.



# Distributing The Custom Step

---

You can distribute a custom step so that users can call the custom step from the executable version of Vision Builder AI or other remote targets.

Complete the following steps to save a custom step for distribution:

1. Launch LabVIEW.
2. Select **Tools»<Vision Builder AI>»Save Your Custom Step for Distribution**.
3. Select the **Manual Threshold** custom step.
4. Select the **Windows PC** checkbox.
5. Click **Build** to save the custom step for distribution.
6. Click **OK** to close the Save Your Custom Step for Distribution wizard.

The Save Your Custom Step for Distribution wizard builds 59 Manual Threshold.bin in the <Vision Builder AI>\UserPlugins folder.

To distribute the custom step to a Windows PC, install 59 Manual Threshold.bin and the corresponding .tif file to the <Vision Builder AI>\UserPlugins folder on the vision inspection system.



**Note** If you choose to build your custom step for one of the remote targets, and choose not to download it right away, you must manually FTP the file to the target. FTP the custom step bin file, located in <Vision Builder AI>\UserPlugins\  
<Target>, to the folder c:\vbai\UserPlugins\ of the remote target.

---

# Logging Measurement and Setting Limit Conditions

This chapter introduces concepts required to create a custom step that performs pass/fail analysis. Complete the steps in this chapter to modify the custom step that you created in Chapter 3, *Creating a Custom Image Processing Step*, to perform the following functions:

- **Log a measurement**—The modified step calculates the percentage of the image that is removed by an image threshold. The percentage is logged and made available to subsequent steps.
- **Set a limit condition**—The modified step compares the logged percentage against a minimum threshold percentage specified through the user interface. Based on the results of the comparison, the step indicates whether the image under inspection passes or fails.



**Note** Refer to the Processing Step that Logs Measurements template as you proceed with this tutorial. Use the Processing Step that Logs Measurements template to create new custom steps that perform pass/fail analysis.



**Note** This tutorial requires the NI Vision Development Module in addition to the minimum system requirements of the Vision Builder AI Development Toolkit.

---

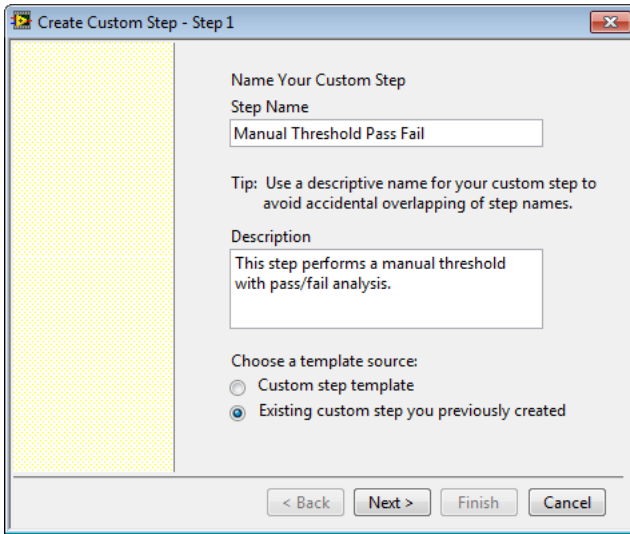
## Generating the Custom Step from an Existing Step

Complete the following steps to generate a new custom step using the Create Custom Step wizard:

1. Launch LabVIEW.
2. Select **Tools»<Vision Builder AI>>Create Custom Step**.
3. Enter `Manual Threshold Pass Fail` as the **Step Name** for the custom step.
4. Enter `This step performs a manual threshold with pass/fail analysis` as the **Description** for the custom step.

5. Select the **Existing custom step you previously created** option. Figure 4-1 illustrates the correct option.

**Figure 4-1.** Creating a Custom Step from an Existing Step



6. Click **Next**.
7. Select **Manual Threshold** from the **Existing Custom Steps Previously Created** list.
8. Click **Next** and verify the setup information. Close all open VIs to ensure that the custom step is created successfully.
9. Click **Finish** to create the custom step.

The Create Custom Step wizard opens the Manual Threshold Pass Fail - All VIs VI. This VI provides a project window that contains the VIs you can modify for the custom step. Refer to the [Source Code VIs](#) section of Chapter 2, [Understanding Custom Steps](#), for information about the VIs that make up a custom step.



**Note** The custom step is not available in Vision Builder AI until you save it for distribution. National Instruments recommends that you debug the custom step before you save it for distribution. Refer to the [Debugging the Custom Step](#) section for information about debugging custom steps.

## Accessing the Custom Step Source VIs

---

The Create Custom Step wizard generates source VIs that provide the functionality of the custom step. The Create Custom Step wizard installs the source VIs to the <Vision Builder AI>\UserPlugins\59 Manual Threshold Pass Fail.llb file. Use the All VIs VI block diagram to access the custom step source VIs.

Complete the following steps to open the All VIs VI for the custom step, if it is not already open:

1. Launch LabVIEW.
2. Select **File»Open**.
3. Navigate to <Vision Builder AI>\UserPlugins\59 Manual Threshold Pass Fail.llb
4. Open the LLB.

Select the Manual Threshold Pass Fail - All VIs.vi and click **OK**.

## Logging Measurements

---

Follow the steps in this section to modify the User Programming VI to perform the following functions:

- Calculate the percentage of the image that is removed by an image threshold.
- Store the thresholded percentage to the Measurements array so that other steps can access the value.
- Log the thresholded percentage so that Vision Builder AI can compare the value with the limits specified by the user in the user interface.

## Calculating the Threshold Percentage

Complete the following steps to modify the User Programming VI so that it calculates the percentage of the image that is removed by an image threshold:

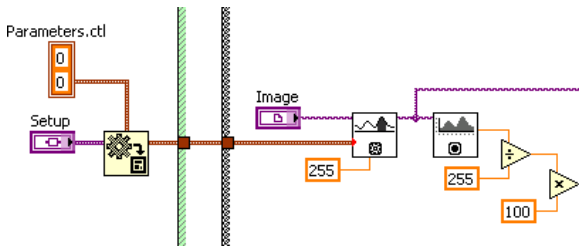
1. Open the All VIs VI.
2. Double-click the User Programming VI to open it.



3. The front panel is never displayed, so it does not have to be modified. Do *not* modify the connector pane of the User Programming VI. Select **Window»Show Block Diagram** to open the block diagram.
4. Navigate to the No Error case of the case structure.
5. Navigate to the Execution case of the case structure within the No Error case.

6. Complete the following steps to modify the block diagram to calculate the percentage of the image removed by the image threshold, as shown in Figure 4-2:

**Figure 4-2.** Calculating the Image Threshold Percentage



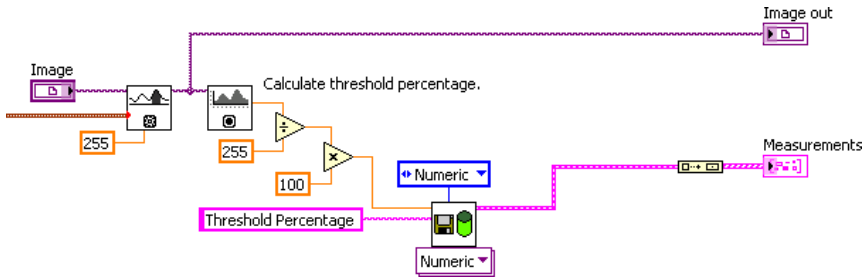
- a. Place an IMAQ Histogram VI inside the execution case of the case structure.
  - b. Connect the **Image Dst Out** output of the IMAQ Threshold VI to the **Image** input of the IMAQ Histogram VI.
  - c. Place a Divide function inside the case structure.
  - d. Connect the **Mean Value** output of the IMAQ Histogram VI to the **x** input of the Divide function.
  - e. Create a numeric constant with a value of 255 and connect it to the **y** input of the Divide function.
  - f. Place a Multiply function inside the Execution case of the case structure.
  - g. Connect the **x/y** output of the Divide function to the **x** input of the Multiply function.
  - h. Create a numeric constant with a value of 100 and connect it to the **y** input of the Multiply function.
7. Select **File»Save**.

The **x\*y** output of the Multiply function provides the percentage of the image removed by the threshold.

## Storing the Threshold Percentage in the Measurements Array

Complete the following steps to modify the block diagram to store the thresholded percentage to the Measurements array so that other steps can access the value, as shown in Figure 4-3:

**Figure 4-3.** Storing the Threshold Percentage in the Measurements Array



1. With the User Programming VI block diagram open, navigate to the Execution case of the case structure within the No Error case.
2. Open the All VIs VI block diagram and locate the VBAI SDK - Create Result VI.



3. Copy the VBAI SDK - Create Result VI into the case structure of the User Programming VI.
4. Connect the **x\*y** output of the Multiply function to the **Result Value** input of the VBAI SDK - Create Result VI. The polymorphic VBAI SDK - Create Result VI automatically switches to the Numeric instance.
5. Right-click the **Type** input and select **Create»Constant** to create a populated enum constant. Select **Numeric** as the type.
6. Create a string constant with a value of `Threshold Percentage` and connect it to the **Name** input of the VBAI SDK - Create Result VI.
7. Place a Build Array function inside the case structure.
8. Connect the **Result** output of the VBAI SDK - Create Result VI to the **element** input of the Build Array function.
9. Locate the Measurements array that is inside the case structure. Connect the **appended array** output of the Build Array function to the Measurements array.
10. Select **File»Save**.

The percentage of the image removed by the threshold is stored in the Measurements array as `Threshold Percentage`.



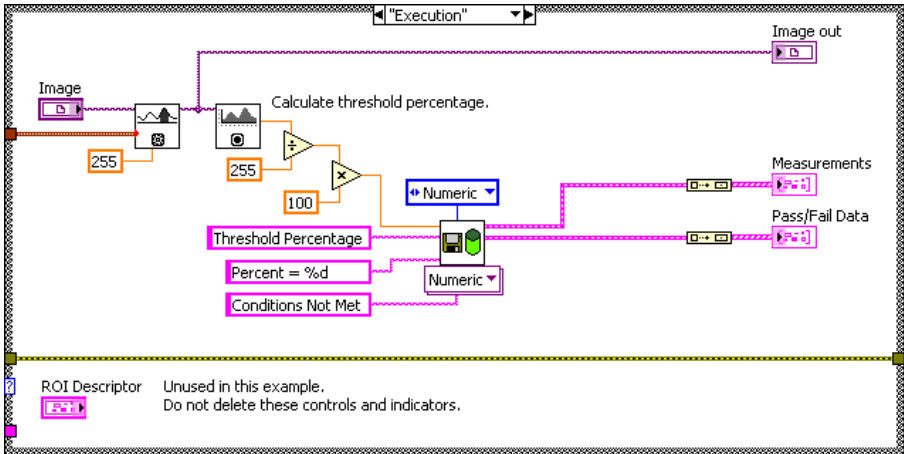
**Note** You can log additional results by adding more elements to the array.

## Returning Pass/Fail Data

The step performs pass/fail analysis by comparing results from the User Programming VI against limit conditions specified by the user in the User Interface VI.

Complete the following steps to modify the block diagram to store the thresholded percentage to the Pass/Fail Data array, as shown in Figure 4-4:

**Figure 4-4.** Storing the Threshold Percentage in the Pass/Fail Data Array



1. With the User Programming VI block diagram open, navigate to the Execution case of the case structure within the No Error case.
2. Create a string constant with a value of `Percent = %d` and connect it to the **Display String** input of the VBAI SDK - Create Result VI.
3. Create a string constant with a value of `Conditions Not Met` and connect it to the **Fail String** input on the bottom of the VBAI SDK - Create Result VI.
4. Place a Build Array function inside the case structure.
5. Connect the **Pass Fail Data** output of the VBAI SDK - Create Result VI to the **element** input of the Build Array function.
6. Locate the Pass/Fail Data array that is inside the case structure. Connect the **appended array** output of the Build Array function to the Pass/Fail Data array.
7. Select **File»Save** to save the User Programming VI.
8. Select **File»Close** to close the User Programming VI.

The percentage of the image removed by the threshold is stored in the Pass/Fail Data array and returned to the User Interface VI so that the step can perform pass/fail analysis and update the step status.



**Note** The **Pass/Fail Data** array must have the same number of elements as the **Pass/Fail Condition out** array found in the User Interface VI. Each element of the **Pass/Fail Data** array is compared to a condition specified in the **Pass/Fail Condition out** array.

## Adding Pass/Fail Controls

---

Complete the following steps to add controls to the User Interface VI that allow the user to enable pass/fail analysis and to specify a limit condition for pass/fail analysis:

1. Open the All VIs VI.
2. Double-click the User Interface VI to open it.



**Note** Do not resize the tab control on the user interface or change its position. Changing the tab control on the custom step user interface might move the user interface controls and indicators to a position that the user cannot access.

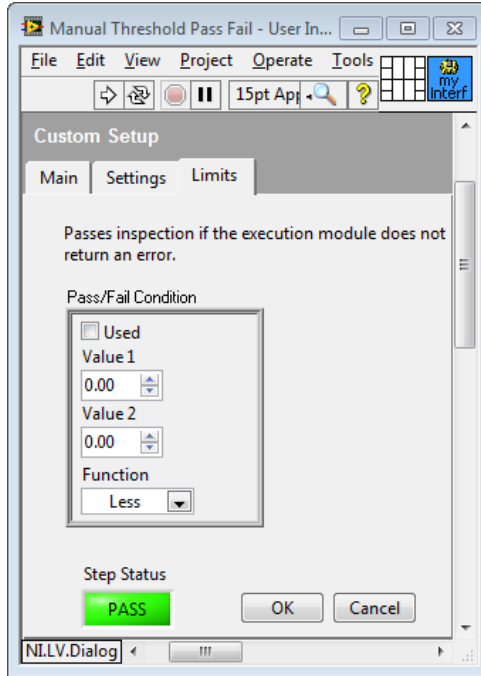
3. Select the **Limits** tab.
4. Right-click the front panel and select **Select a Control**.
5. Navigate to `<Vision Builder AI>\Plugins\Common SDK VIs.llb`, select `VBAI SDK - Pass Fail Condition.ctl`, and click **OK**.

The Pass/Fail Condition control is a type definition that includes the controls necessary to enable and perform pass/fail analysis.



- Place the Pass/Fail Condition control on the Limits tab, as illustrated in Figure 4-5.

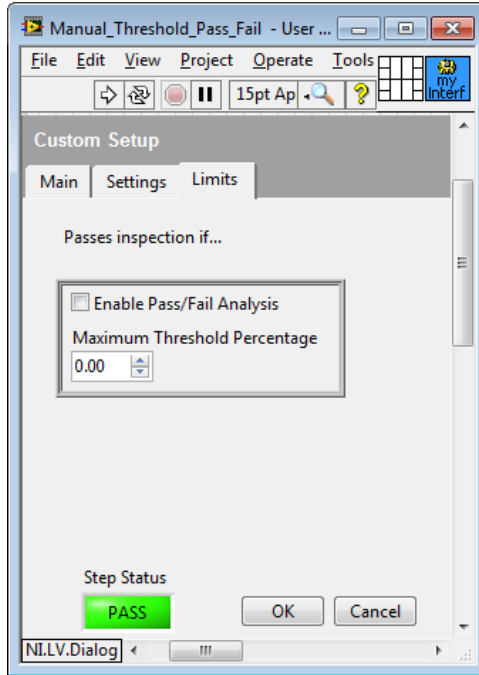
**Figure 4-5.** Adding Controls for Pass/Fail Analysis



- Right-click the Used Boolean and select **Visible Items»Caption**. Enter Enable Pass/Fail Analysis as the caption text.
- Right-click the Value 1 control and select **Visible Items»Caption**. Enter Maximum Threshold Percentage as the caption text.

- The Pass/Fail Condition cluster contains controls that the user should never delete. Resize the Pass/Fail Condition cluster to hide the Value 2 and Function controls, as illustrated in Figure 4-6.

**Figure 4-6.** Resizing the Pass/Fail Condition Cluster



- Select **File»Save**.

The user interface now contains controls that allow the user to enable pass/fail analysis, and to specify a limit condition for pass/fail analysis.

# Initializing Values for the Pass/Fail Controls

Complete the steps in this section to initialize the pass/fail controls with correct values.

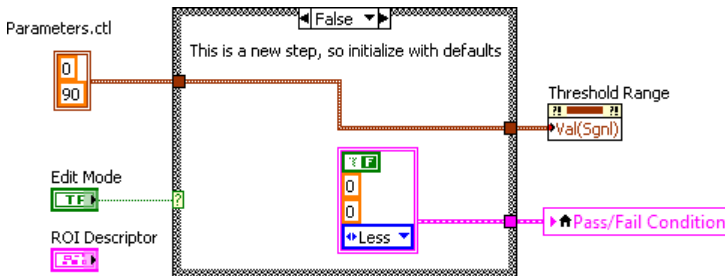
## Initializing Default Values for the Pass/Fail Controls

When user adds a custom step to the inspection, the step must initialize its controls to default values.

Complete the following steps to modify the User Interface VI block diagram to initialize the pass/fail controls with default values:

1. With the User Interface VI front panel open, select the Limits tab.
2. Right-click the Pass/Fail Condition cluster and select **Find»Terminal**. LabVIEW highlights the Pass/Fail Condition control on the User Interface block diagram.
3. Move the Pass/Fail Condition control outside of the stacked sequence structure. Do not delete this control.
4. Select frame 1 of the stacked sequence structure. Locate the case structure within frame 1 and select the **False** case. The false case specifies default values for the control.
5. Complete the following steps to build the block diagram shown in Figure 4-7:

**Figure 4-7.** Initializing the Pass/Fail Controls with Default Values



- a. Place a local variable outside the right side of the case structure.
  - b. Click the local variable and select **Pass/Fail Condition** from the list.
  - c. Right-click the **Pass/Fail Condition** variable and select **Create»Constant**.
  - d. Place the constant inside the case structure.
  - e. Connect the constant to the Pass/Fail Condition variable.
6. Select **File»Save**.

When the user adds the custom step to an inspection in Vision Builder AI, the User Interface VI initializes the pass/fail controls to the values specified in the constant.

## Saving Values for the Pass/Fail Controls

When user closes the step, the step must save the value specified by the user so that the step can reinitialize the user-specified values if the user edits the step.

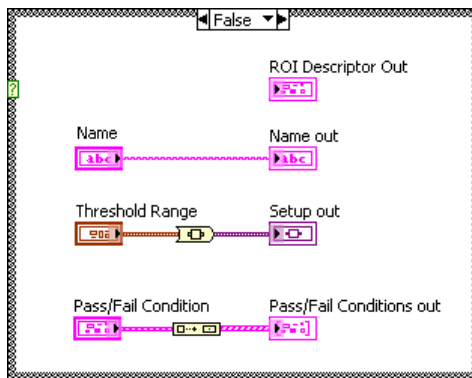
Complete the following steps to modify the block diagram to save the values specified by the user:

1. With the User Interface VI block diagram open, select frame 4 of the stacked sequence structure. Locate the case structure within frame 4 and select the **False** case.

The False case stores the step values in the indicators so that the values can be passed to the Vision Builder AI engine. When a user edits the step, the Vision Builder AI engine passes the stored values back to the step.

2. Complete the following steps to build the block diagram shown in Figure 4-8:

**Figure 4-8.** Saving the Pass/Fail Condition Control Values



- a. Place the **Pass/Fail Condition** control inside the case structure.
- b. Place a Build Array function inside the case structure.
- c. Connect the Pass/Fail Condition control to the **element** input of the Build Array function.
- d. Disconnect the **Pass/Fail Conditions in** array from the **Pass/Fail Conditions out** array. Move the **Pass/Fail Conditions in** array outside of the stacked sequence structure. Do not delete this array.
- e. Connect the output of the Build Array function to the **Pass/Fail Conditions out** array.

When the user saves the step, the values from the **Pass/Fail Condition** control are stored in the **Pass/Fail Conditions out** array. When the user edits the step, the Vision Builder AI engine passes the values stored in the **Pass/Fail Conditions out** array to the **Pass/Fail Conditions in** array. Use the **Pass/Fail Conditions in** array to initialize values for the pass/fail controls on the Limits tab.

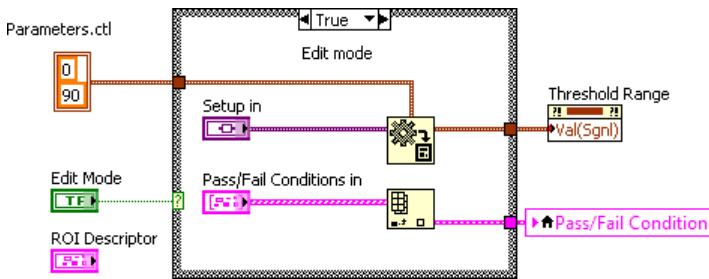
## Initializing Existing Values for the Pass/Fail Controls

When a user edits a custom step the step must initialize its controls to the values last specified by the user.

Complete the following steps to modify the User Interface VI block diagram to initialize the pass/fail controls with existing values:

1. With the User Interface VI block diagram open, select frame 1 of the stacked sequence structure.
2. Locate the case structure within frame 1 and select the **True** case. The true case handles existing values for the control.
3. Complete the following steps to build the block diagram shown in Figure 4-9:

**Figure 4-9.** Initializing the Pass/Fail Controls with Existing Values



- a. Place the **Pass/Fail Conditions in** control inside the case structure in frame 1.
  - b. Place an **Index Array** function inside the case structure.
  - c. Connect the **Pass/Fail Conditions in** control to the **array** input of the **Index Array** function.
  - d. Connect the **element** output of the **Index Array** function to the **Pass/Fail Condition** variable.
4. Select **File»Save**.

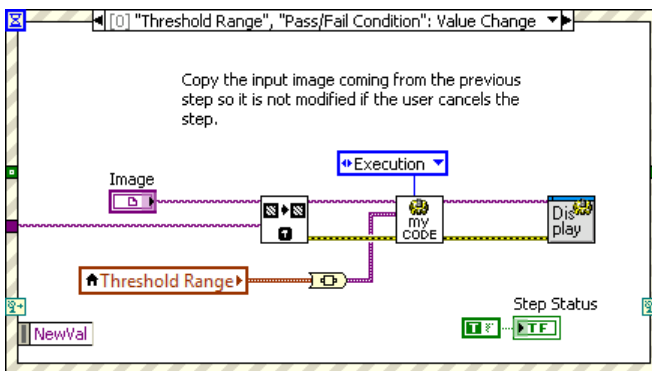
When the user edits the step, the User Interface VI initializes the pass/fail controls to the values specified in the **Pass/Fail Conditions in** array.

## Handling Value Changes for the Pass/Fail Controls

Complete the following steps to update the User Interface VI block diagram to handle changes to the pass/fail controls:

1. With the User Interface VI block diagram open, select frame 3 of the stacked sequence structure.
2. Locate the event structure within frame 3 and select event case 0. Event case 0 handles changes to the threshold range controls.
3. Complete the following steps to update event case 0 to handle changes to the pass/fail controls:
  - a. Right-click on the event structure and select **Edit Events Handled by This Case**.
  - b. Locate the **Event Specifiers** list and click **Add Event**.
  - c. In the **Event Sources** list, expand the **Pass/Fail Condition** entry and select **<All Elements>**.
  - d. In the **Events** list select **Value Change**.
  - e. Click **OK** to close the Edit Events dialog box.
4. Disconnect the Event Data Node from the To Variant function. Because event case 0 now handles multiple events, you must modify the source code that handles changes to the threshold range controls.
5. Place a local variable inside the event structure.
6. Click the local variable and select **Threshold Range** from the list.
7. Right-click the Threshold Range variable and select **Change To Read**.
8. Connect the Threshold Range variable to the **anything** input of the To Variant function. Figure 4-10 illustrates the modified event structure.

**Figure 4-10.** Reconnecting the Threshold Range Control

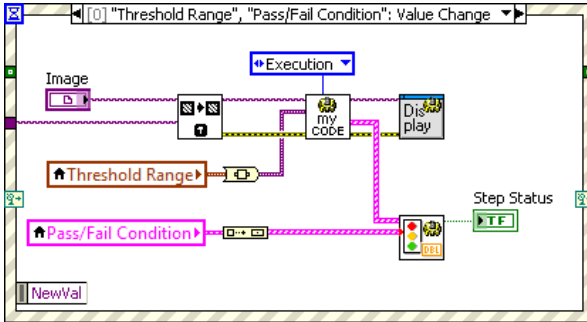



9. Select **File»Save** to save the User Interface VI.

# Setting the Step Status Based on Pass/Fail Analysis

Complete the following steps to modify the User Interface VI block diagram to update the status of the step based on pass/fail analysis, as shown in Figure 4-11:

**Figure 4-11.** Setting the Step Status Based on Pass/Fail Analysis



1. With the User Interface VI block diagram open, select frame 3 of the stacked sequence structure.
  2. Locate the event structure within frame 3 and select event case 0.
  3. Place a local variable inside the event structure.
  4. Click the local variable and select **Pass/Fail Condition** from the list.
  5. Right-click the Pass/Fail Condition variable and select **Change to Read**.
  6. Place a **Build Array** function inside the event structure.
  7. Connect the **Pass/Fail Condition** variable to the **element** input of the Build Array function.
  8. Open the All VIs VI block diagram and locate the VBAI Decision Maker (Float) VI.
  9. Copy the **VBAI Decision Maker (Float)** VI into the event structure of the User Interface VI.
- 
10. The VBAI Decision Maker (Float) VI compares results from User Programming VI against the limit conditions specified by the user in the User Interface VI to determine whether the step result conditions are met.
  11. Connect the output of the Build Array function to the **Pass/Fail Condition** input of the VBAI Decision Maker (Float) VI.
  12. Connect the **Pass/Fail Data** output of the Manual Threshold Pass Fail - User Programming VI to the **Pass/Fail Data** input of the VBAI Decision Maker (Float) VI.

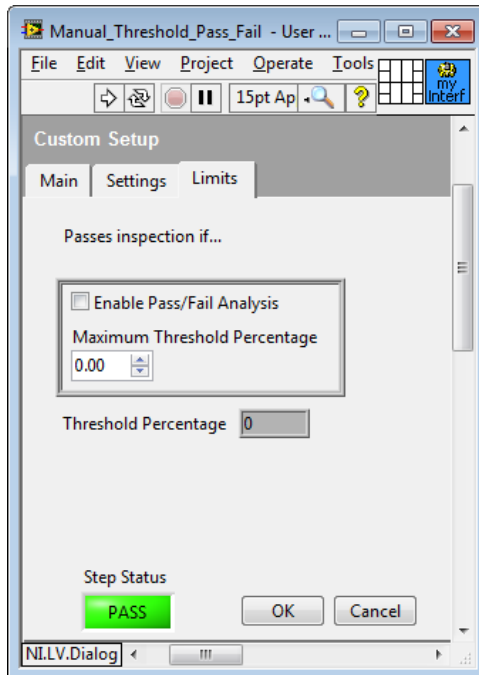
13. Delete the Boolean constant that is connected to the Step Status indicator.
14. Connect the **Pass/Fail Flag** output of the VBAI Decision Maker (Float) VI to the Step Status indicator.
15. Select **File»Save** to save the User Interface VI.

## Updating the User Interface with Measurement Data

Complete the following steps to update the User Interface VI to display measurement data from the User Programming VI:

1. With the User Interface block diagram open, select frame 3 of the stacked sequence structure.
2. Locate the event structure within frame 3 and select event case 0.
3. Press <Ctrl-E> or select **Window»Show Front Panel** to open the front panel.
4. Select the **Limits** tab.
5. Place a numeric indicator on the Limits tab. Label the numeric indicator `Threshold Percentage` as illustrated in Figure 4-12.

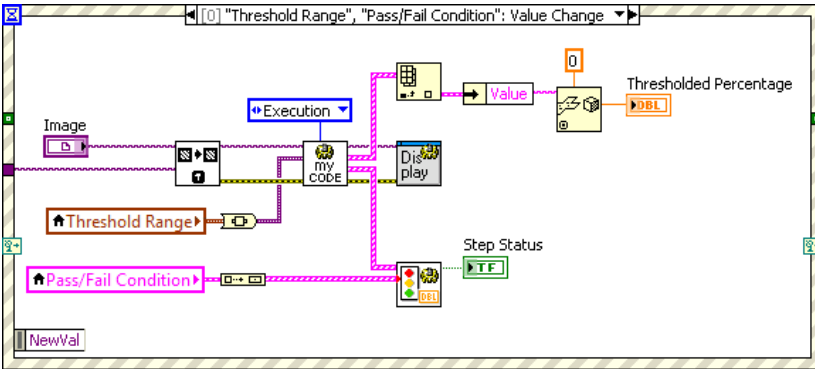
**Figure 4-12.** Adding a Threshold Percentage Indicator to the Limits Tab





6. Right-click the Threshold Percentage indicator and select **Properties**.
7. In the **Appearance** tab, select the **Disabled** option. Click **OK** to close the properties dialog box.
8. Press <Ctrl-E> or select **Window»Show Block Diagram** to return to the block diagram.
9. Complete the following steps to build the block diagram show in Figure 4-13:

**Figure 4-13.** Updating the User Interface with Measurement Data



- a. Place an Index Array function inside the event structure.
  - b. Connect the **Measurements** output of the Manual Threshold Pass Fail - User Programming VI to the **array** input of the Index Array function.
  - c. Place an Unbundle By Name function inside the event structure.
  - d. Connect the **element** output of the Index Array function to the Unbundle By Name function.
  - e. Place an Unflatten From String function inside the event structure.
  - f. Connect the **Value** output of the Unbundle By Name function to the **binary string** input of the Unflatten From String function.
  - g. Place a Numeric constant inside the event structure and connect it to the **type** input of the Unflatten From String function.
  - h. Right-click the Numeric constant and select **Representation»Double Precision**.
  - i. Connect the **value** output of the Unflatten From String function to the Threshold Percentage indicator. Figure 4-13 illustrates the complete block diagram.
10. Select **File»Save** to save the User Interface VI.
  11. Select **File»Close** to close the User Interface VI.

The Manual Threshold Pass Fail step is complete.

# Debugging the Custom Step

---

Complete the following steps to debug a custom step:

1. Launch LabVIEW.



**Note** You must configure LabVIEW before debugging a custom step. Refer to the [Configuring LabVIEW to Debug Custom Steps](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for more information about configuring LabVIEW.

2. Select **Tools»<Vision Builder AI >>Test Your Custom Step in Vision Builder AI** to launch Vision Builder AI and test the step.



**Note** You must open Vision Builder AI from LabVIEW to test a custom step that has not been saved for distribution. Refer to the [Distributing The Custom Step](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for information about saving custom steps for use with the Vision Builder AI executable.

3. Click **Configure Inspection** to open the Vision Builder AI Configuration interface.
4. Add a **Simulate Acquisition** step to the acquisition. Because the Manual Threshold Pass Fail custom step operates on an image, you must acquire an image to test the custom step.
5. Add the Manual Threshold Pass Fail custom step to the inspection. By default, custom steps are located on the **Use Additional Tools** palette.

If you did not modify the custom step icon, the custom step displays the default custom step icon.



6. When you open the custom step, Vision Builder AI operates in debugging mode. The front panel of the User Interface VI for the custom step opens in LabVIEW. The front panel of the User Interface VI may open behind the Vision Builder AI window. Open the User Interface VI by clicking the VI window in the taskbar.
7. Complete the following steps to debug the custom step:
  - a. Select the **Limits** tab of the Manual Threshold Pass Fail custom step.
  - b. Locate the Thresholded Percentage indicator and note the value.
  - c. Select the **Enable Pass/Fail Analysis** checkbox. Note that the Step Status indicator changes to FAIL. The step fails because the thresholded percentage exceeds the specified maximum threshold percentage value of 0.
  - d. Enter 20 as the value for the **Maximum Threshold Percentage** control. Note that the Step Status indicator changes to PASS.
  - e. Select the **Settings** tab.

- f. Enter 200 as the value for the Maximum Value control. Note that the Step Status indicator changes to FAIL because the thresholded percentage once again exceeds the minimum required threshold percentage.
- g. Click **OK** to add the step to the inspection, then edit the step to verify that the controls initialize to the correct values.

When you are satisfied with the performance of the step, save the step for distribution. Refer to the *Distributing The Custom Step* section of Chapter 3, *Creating a Custom Image Processing Step*, for more information about saving custom steps for distribution.

---

# Using Previous Measurements

Complete the steps in this chapter to modify the custom step that you created in Chapter 4, *Logging Measurement and Setting Limit Conditions*, so that the user can select measurement results from previous steps as threshold range values.

You can use the Measurements indicator in the Execution mode of the User Programming VI to log double, Boolean, and string measurements of the custom step for use in subsequent steps. For more information about the Measurements indicator, refer to the *User Programming VI Parameters* section of Appendix A, *Controls and Indicators Used in Source Code VIs*.



**Note** This tutorial requires the NI Vision Development Module in addition to the minimum system requirements of the Vision Builder AI Development Toolkit.

---

## Generating the Custom Step from an Existing Step

Complete the following steps to generate a new custom step using the Create Custom Step wizard:

1. Launch LabVIEW.
2. Select **Tools»<Vision Builder AI>>Create Custom Step**.
3. Enter `Manual Threshold Previous Measurements` as the **Step Name** for the custom step.
4. Enter `This step performs a threshold based on previous measurements as the Description` for the custom step.
5. Select the **Existing custom step you previously created** option.
6. Click **Next**.
7. Select **Manual Threshold Pass Fail** from the **Existing Custom Steps Previously Created** list.
8. Click **Next** and verify the setup information. Close all open VIs to ensure that the custom step is created successfully.
9. Click **Finish** to create the custom step.

The Create Custom Step wizard opens the Manual Threshold Previous Measurements - All VIs VI. This VI provides a project window that contains the VIs you can modify for the custom step.



**Note** The custom step is not available in Vision Builder AI until you save it for distribution. National Instruments recommends that you debug the custom step before you save it for distribution. Refer to the [Debugging the Custom Step](#) section for information about debugging custom steps.

## Accessing the Custom Step Source VIs

---

The Create Custom Step wizard generates source VIs that provide the functionality of the custom step. The Create Custom Step wizard installs the source VIs to the <Vision Builder AI>\UserPlugins\59 Manual Threshold Previous Measurements.llb file. Use the All VIs VI block diagram to access the custom step source VIs.

Complete the following steps to open the All VIs VI for the custom step, if it is not already open:

1. Launch LabVIEW.
2. Select **File»Open**.
3. Navigate to <Vision Builder AI>\UserPlugins\59 Manual Threshold Previous Measurements.llb
4. Open the LLB.
5. Select the Manual Threshold Previous Measurements - All VIs.vi and click **OK**.

## Modifying the Type Definition

---

You must update the custom step type definition to use measurements from previous steps. Vision Builder AI steps store measurement information as a globally unique identifier (GUID). A GUID is a cluster that contains two strings and a numeric value that identifies the measurement.

Complete the following steps to add two GUIDs to the type definition:

1. Open the All VIs VI.
2. Double-click the User Interface VI to open it.



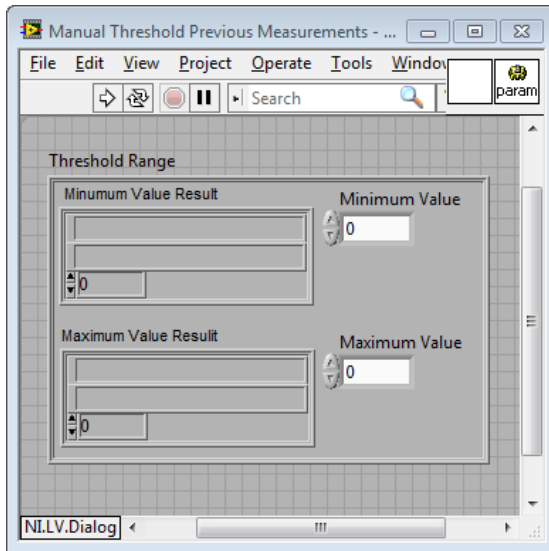
3. Select the **Settings** tab.



**Note** Do not resize the tab control on the user interface or change its position. Changing the tab control on the custom step user interface might move the user interface controls and indicators to a position that the user cannot access.

4. Right-click the Threshold Range cluster and select **Open Type Def** to open the Parameters control.
5. Expand the size of the Parameters control window and the Threshold Range cluster.
6. Complete the following steps to build the front panel shown in Figure 5-1:

**Figure 5-1. Updating the Type Definition**



- a. Right-click the front panel and select **Select a Control**.
  - b. Navigate to <Vision Builder AI>\Plugins\Common SDK VIs.llb, select IVB Measure ID.ct1, and click **OK**.
  - c. Place the control inside the Threshold Range cluster. Label the control Minimum Value Result.
  - d. Select the **Minimum Value Result** cluster.
  - e. Select **Edit>Copy** to copy the cluster.
  - f. Click the background of the Threshold Range cluster, then select **Edit>Paste** to paste a copy of the **Minimum Value Result** cluster.
  - g. Label the new cluster Maximum Value Result.
7. Select **File>Save** to save the type definition.
  8. Select **File>Close** to close the type definition.

# Adding Previous Measurement Controls

Complete the following steps to add controls to the User Interface VI to allow the user to select threshold range values from a list of result measurements:

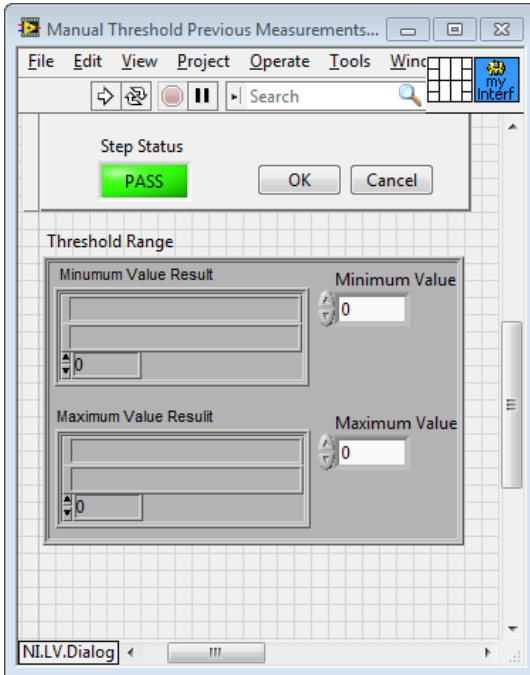
1. With the User Interface VI front panel open, select the **Settings** tab.
2. Expand the User Interface VI front panel window.



**Note** Do not resize the tab control on the user interface or change its position. Changing the tab control on the custom step user interface might move the user interface controls and indicators to a position that the user cannot access.

3. Move the **Threshold Range** cluster off of the tab control, as illustrated in Figure 5-2:

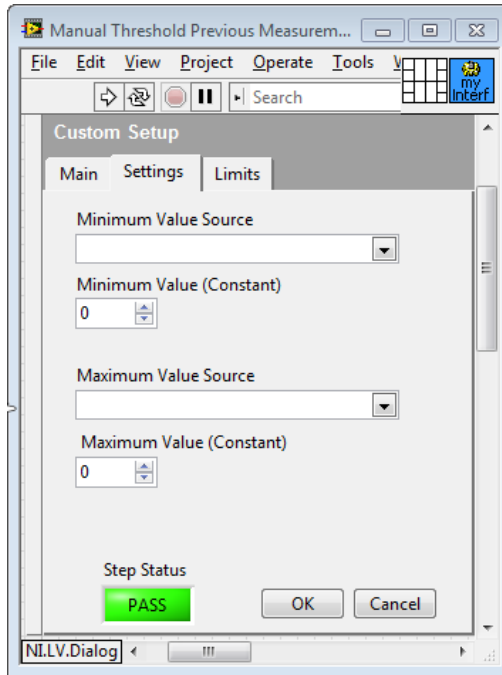
**Figure 5-2.** Moving the Threshold Range Cluster



**Tip** Position hidden controls to the right of the tab control in a column or row. Controls placed below the tab control may be visible if the Vision Builder AI window is maximized.

4. Complete the following steps to build the front panel shown in Figure 5-3:

**Figure 5-3.** Adding Previous Measurements Controls



- a. Add a system ring to the Settings tab. Label the system ring *Minimum Value Source*.
  - b. Add a system ring to the Settings tab. Label the system ring *Maximum Value Source*.
  - c. Add a system spin control to the Settings tab. Label the system spin control *Minimum Value (Constant)*.
  - d. Add a system spin control to the Settings tab. Label the system spin control *Maximum Value (Constant)*.
5. Press <Ctrl-E> or select **Window»Show Block Diagram** to show the block diagram.
  6. Move the *Minimum Value Source*, *Minimum Value (Constant)*, *Maximum Value Source*, and *Maximum Value (Constant)* controls outside the stacked sequence structure.
  7. Select **File»Save**.

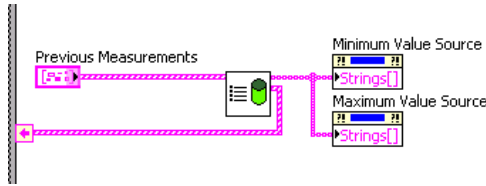


# Enumerating Previous Measurements

Complete the following steps to modify the User Interface VI to populate the previous measurement controls with a list of previous measurements:

1. With the User Interface VI block diagram open, select frame 0 of the stacked sequence structure.
2. Complete the following steps to build the block diagram illustrated in Figure 5-4:

**Figure 5-4.** Enumerating a List of Previous Measurements



- a. Locate the Previous Measurements array. Move the Previous Measurements array to the top of the stacked sequence structure.
- b. Open the All VIs VI block diagram and locate the VBAI SDK - Populate Result Ring VI.

The VBAI SDK - Populate Result Ring VI loops through the results returned by previous steps in the inspection and returns a list of results and corresponding GUIDs of the selected data type. The default data type is numeric.

- c. Copy the VBAI SDK - Populate Result Ring VI into the stacked sequence structure of the User Interface VI.



- d. Connect the Previous Measurements array to the **Previous Measurements** input of the VBAI SDK - Populate Results Ring VI.
- e. Right-click the Minimum Value Source ring control and select **Create»Property Node»Strings[]**.
- f. Place the Minimum Value Source property node inside the stacked sequence structure.
- g. Select the Minimum Value Source property node and select **Change to Write**.
- h. Right-click the Maximum Value Source ring control and select **Create»Property Node»Strings[]**.
- i. Place the Maximum Value Source property node inside the stacked sequence structure.
- j. Select the Maximum Value Source property node and select **Change to Write**.
- k. Connect the **Previous Measurement Names** output of the VBAI SDK - Populate Results Ring VI to the **Strings[]** input of the Minimum Value Source and Maximum Value Source property nodes.

1. Right-click the border of the stacked sequence structure and select **Add Sequence Local**.
  - m. Place a free label next to the sequence local, outside the stacked sequence structure. Enter `Previous Measurement IDs` as the label text.
  - n. Connect the **Measurement IDs** output of the VBAI SDK - Populate Results Ring VI to the sequence local.
3. Select **File»Save**.

## Initializing Values for Previous Measurement Controls

Complete the steps in this section to initialize the previous measurement controls with correct values.

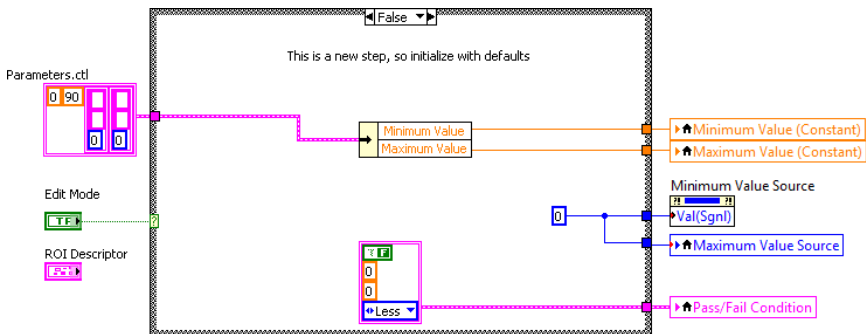
## Initializing Default Values for Previous Measurement Controls

When the user adds a custom step to the inspection, the step must initialize its controls to default values.

Complete the following steps to modify the User Interface VI block diagram to initialize the previous measurement controls with default values:

1. With the User Interface VI block diagram open, select frame 1 of the stacked sequence structure.
2. Locate the case structure within frame 1 and select the **False** case. The false case specifies default values for the control.
3. Complete the following steps to build the block diagram shown in Figure 5-5:

**Figure 5-5.** Initializing the Previous Measurement Controls with Default Values



- a. Place an Unbundle By Name function inside the case structure.
- b. Connect the Parameters.ctl cluster to the input of the Unbundle By Name structure.

- c. Expand the Unbundle By Name function so that two elements are visible.
  - d. Right-click the Minimum Value (Constant) control and select **Create»Local Variable**.
  - e. Place the Minimum Value (Constant) variable inside the stacked sequence structure.
  - f. Connect the **Minimum Value** output of the Unbundle By Name function to the input of the Minimum Value (Constant) variable.
  - g. Right-click the Maximum Value (Constant) control and select **Create»Local Variable**.
  - h. Place the Maximum Value (Constant) variable inside the stacked sequence structure.
  - i. Connect the **Maximum Value** output of the Unbundle By Name function to the input of the Maximum Value (Constant) variable.
  - j. Locate the Threshold Range property node to the right of the case structure. Right-click the Threshold Range property node and select **Link to»Pane»Tab Control»Minimum Value Source**.
  - k. Right-click the Maximum Value Source control and select **Create»Local Variable**.
  - l. Place the Maximum Value variable inside the stacked sequence structure.
  - m. Place a numeric constant inside the case structure.
  - n. Connect the numeric constant to the Minimum Value Source property node and the Maximum Value Source variable.
4. Select **File»Save**.

When the user adds the custom step to an inspection in Vision Builder AI, the User Interface VI initializes the Minimum Value Constant and Maximum Value Constant controls to default values. The User Interface VI initializes the Minimum Value Source and Maximum Value Source ring structures to a value of 0, which corresponds to a selection of the constant values as the source values.

## Initializing Existing Values for Previous Measurements Controls

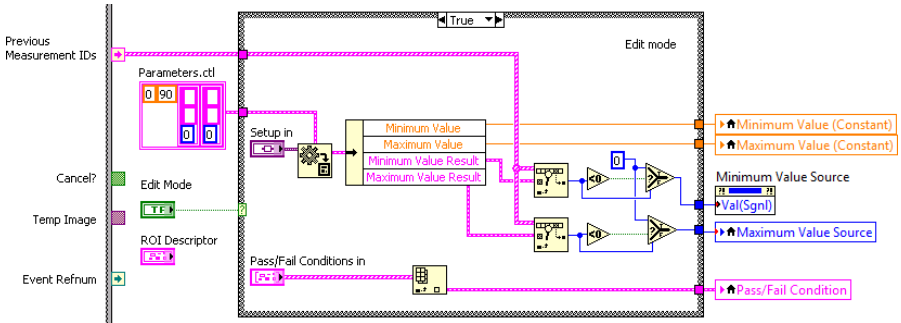
When user edits a custom step the step must initialize its controls to the values last specified by the user.

Complete the following steps to modify the User Interface VI block diagram to initialize the previous measurement controls with default values:

1. With the User Interface VI block diagram open, select frame 1 of the stacked sequence structure.
2. Locate the case structure within frame 1 and select the **True** case. The true case handles existing values for the control.

3. Complete the following steps to build the block diagram shown in Figure 5-6:

**Figure 5-6.** Initializing the Previous Measurement Controls with Existing Values



- a. Place an Unbundle By Name function inside the case structure.
- b. Connect the **data** output of the Variant To Data function to the input of the Unbundle By Name structure.
- c. Expand the Unbundle By Name function so that four elements are visible.
- d. Connect the **Minimum Value** output of the Unbundle By Name function to the Minimum Value (Constant) variable.
- e. Connect the **Maximum Value** output of the Unbundle By Name function to the Maximum Value (Constant) variable.
- f. Place a Search 1D Array function inside the case structure.
- g. Connect the **Previous Measurement IDs** sequence local to the **1D array** input of the Search 1D Array function.
- h. Connect the **Minimum Value Result** output of the Unbundle By Name function to the **element** input of the Search 1D Array function.
- i. Place a Less Than 0? function inside the case structure.
- j. Connect the **index of elements** output of the Search 1D Array function to the **x** input of the Less Than 0? function.
- k. Place a Select function inside the case structure.
- l. Connect the **x < 0?** output of the Less Than 0? function to the **s** input of the Select function.
- m. Place a numeric constant inside the case structure.
- n. Connect the numeric constant to the **t** input of the Select function.
- o. Connect the **index of elements** output of the Search 1D Array function to the **f** input of the Select function.
- p. Connect the **s? t:f** output of the Select function to the Minimum Value Source property node.
- q. Place a Search 1D Array function inside the case structure.

- r. Connect the **Previous Measurement IDs** sequence local to the **ID array** input of the Search ID Array function.
  - s. Connect the **Maximum Value Result** output of the Unbundle By Name function to the **element** input of the Search ID Array function.
  - t. Place a Less Than 0? function inside the case structure.
  - u. Connect the **index of elements** output of the Search ID Array function to the **x** input of the Less Than 0? function.
  - v. Place a Select function inside the case structure.
  - w. Connect the **x < 0?** output of the Less Than 0? function to the **s** input of the Select function.
  - x. Connect the numeric constant to the **t** input of the Select function.
  - y. Connect the **index of elements** output of the Search ID Array function to the **f** input of the Select function.
  - z. Connect the **s? t:f** output of the Select function to the Maximum Value Source variable.
4. Select **File»Save**.

When the user edits the step, the Vision Builder AI engine passes that values stored in the **Setup out** array to the **Setup in** array. The User Interface VI initializes the previous measurement controls to the values specified in the **Setup in** array.

The User Interface VI uses the Search ID Array functions to verify that the values stored for the Minimum Value Source and Maximum Value Source ring structures correspond to the Measurement IDs of available measurements. If the search function returns an index with a negative value, then the measurement is missing and the corresponding ring structure is initialized to a value of 0.

## Handling Value Changes for Previous Measurement Controls

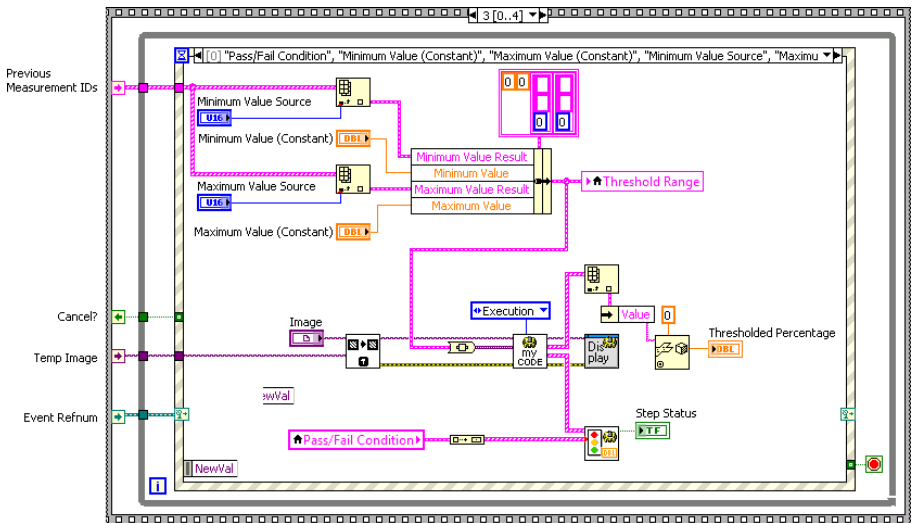
---

Complete the following steps to modify the User Interface VI to handle value changes for the controls available to the user:

1. With the User Interface VI block diagram open, select frame 3 of the stacked sequence structure.
2. Locate the event structure within frame 3 and select event case 0. Event case 0 handles changes to the threshold range controls and changes to the pass/fail controls.
3. Right-click the border of the event structure and select **Edit Events Handled by This Case** to open the Edit Events dialog box.
4. Complete the following instructions to modify the events that event case 0 handles:
  - a. In the **Event Specifiers** list, select **Threshold Range**.
  - b. Click **Remove** to remove Threshold Range from the Event Specifiers list.

- c. Click **Add Event**.
  - d. In the **Event Sources** list select **Minimum Value Source**.
  - e. In the **Events** list select **Value Change**.
  - f. Click **Add Event**.
  - g. In the **Event Sources** list select **Maximum Value Source**.
  - h. In the **Events** list select **Value Change**.
  - i. Click **Add Event**.
  - j. In the **Event Sources** list select **Minimum Value (Constant)**.
  - k. In the **Events** list select **Value Change**.
  - l. Click **Add Event**.
  - m. In the **Event Sources** list select **Maximum Value (Constant)**.
  - n. In the **Events** list select **Value Change**.
  - o. Click **OK** to close the Edit Events dialog box.
5. Place a Bundle By Name function inside the event structure. Expand the Bundle By Name function to display four elements. Complete the following steps to configure the Bundle By Name function as illustrated in Figure 5-7:

**Figure 5-7. Handling Previous Measurement Selections**



- a. Locate the Threshold Range variable. Disconnect the Threshold Range variable from the To Variant function and move the Threshold Range variable to the right of the Bundle By Name function.
- b. Right-click the Threshold Range Variable and select **Create»Constant**.

- c. Connect the constant to the **input cluster** input of the Bundle By Name function.
  - d. Click element 0 of the Bundle By Name function and select **Minimum Value Result»All Elements**.
  - e. Click element 1 of the Bundle By Name function and select **Minimum Value**.
  - f. Click element 2 of the Bundle By Name function and select **Maximum Value Result»All Elements**.
  - g. Click element 3 of the Bundle By Name function and select **Maximum Value**.
6. Complete the following steps to build the block diagram illustrated in Figure 5-7:
- a. Move the Minimum Value Source control inside the event structure.
  - b. Place an Index Array function inside the event structure.
  - c. Connect the Minimum Value Source control to the **index** input of the Index Array function.
  - d. Connect the **Previous Measurement IDs** sequence local to the **array** input of the Index Array functions.
  - e. Connect the **element** output of the Index Array function to the **Minimum Value Result** input of the Bundle function.
  - f. Move the Minimum Value (Constant) control inside the event structure.
  - g. Connect the Minimum Value (Constant) control to the **Minimum Value** input of the Bundle function.
  - h. Move the Maximum Value Source control inside the event structure.
  - i. Place an Index Array function inside the event structure.
  - j. Connect the Maximum Value Source control to the **index** input of the Index Array function.
  - k. Connect the **Previous Measurement IDs** sequence local to the **array** input of the Index Array functions.
  - l. Connect the **element** output of the Index Array function to the **Maximum Value Result** input of the Bundle function.
  - m. Move the Maximum Value (Constant) control inside the event structure.
  - n. Connect the Maximum Value (Constant) control to the **Maximum Value** input of the Bundle function.
  - o. Right-click the Threshold Range variable and select **Change to Write**.
  - p. Connect the **output cluster** output of the Bundle By Name function to the Threshold Range variable.
  - q. Connect the **output cluster** output of the Bundle function to the **anything** input of the To Variant function.
7. Select **File»Save**.

## Handling a New Image

---

Complete the following steps to modify the custom step to run when the inspection image changes:

1. With the User Interface VI block diagram open, select frame 3 of the stacked sequence structure.
2. Locate the event structure within frame 3 and select event case 2. Event case 2 handles changes to the image under inspection.
3. Locate the case structure within event case 2 and select the **New Image** case. When the image changes, for example when the user clicks the Run Once button, the New Image case updates the property nodes within the case. The property node update triggers the User Programming VI, and causes the step to run for the current image.

Because the Threshold Range cluster is no longer handled by the event structure, updating the properties of the threshold range no longer triggers the User Programming VI.

4. Right-click the Value property node and select **Link to»Pane»Tab Control»Minimum Value (Constant)**.
5. Right-click the Val(Signl) property node and select **Link to»Pane»Tab Control»Minimum Value (Constant)**.
6. Select **File»Save** to save the User Interface VI.
7. Select **File»Close** to close the User Interface VI.

When the user clicks the Run Once button, the User Interface VI updates the property nodes for the Minimum Value (Constant) control and triggers the User Programming VI.

## Performing an Image Threshold with Previous Measurement

---

Complete the following steps to modify the User Programming VI to perform an image threshold using previous measurement values:

1. Open the All VIs VI.
2. Double-click the User Programming VI to open it.

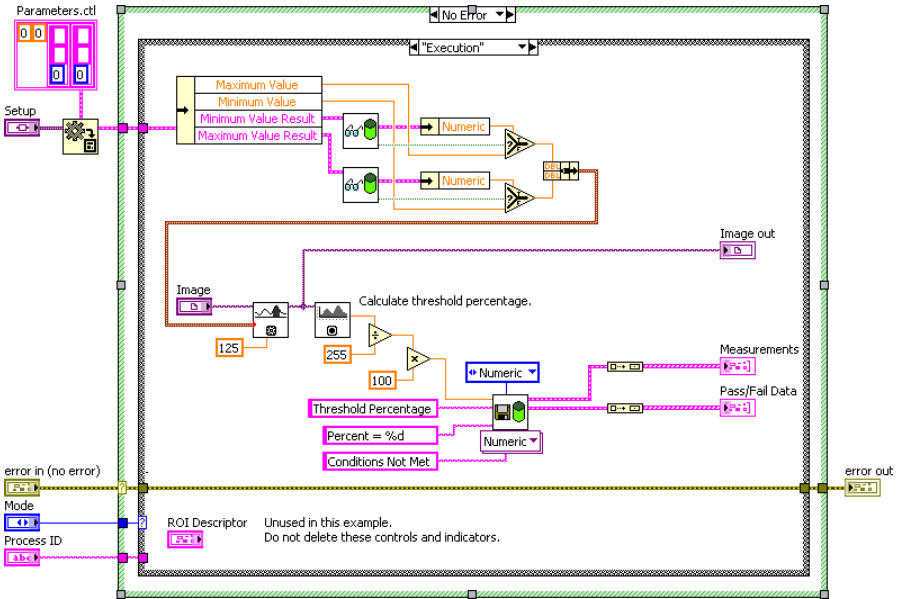


3. Select **Window»Show Block Diagram** to open the block diagram.
4. Navigate to the No Error case of the case structure.
5. Navigate to the Execution case of the case structure within the No Error case.



6. Complete the following steps to build the block diagram illustrated in Figure 5-8:

**Figure 5-8.** Verifying the Validity of Previous Measurement Values



- a. Delete the broken wire.
- b. Place an Unbundle By Name function inside the case structure. Expand the Unbundle By Name function so that four elements are visible.
- c. Connect the **data** output of the Variant To Data function to the **input cluster** input of the Unbundle By Name function.
- d. Open the All VIs VI block diagram and locate the VBAI SDK Get Result VI.
- e. Place a VBAI SDK Get Result VI inside the case structure.



- f. Connect the **Minimum Value Result** output of the Unbundle By Name function to the **Measure ID** input of the VBAI SDK Get Result VI.
- g. Place an Unbundle By Name function inside the case structure.
- h. Connect the **Result Value** output of the VBAI SDK Get Result VI to the **input cluster** input of the Unbundle By Name function.
- i. Place a Select function inside the case structure.
- j. Connect the **Result Available** output of the VBAI SDK Get Result VI to the **s** input of the Select function.

- k. Connect the **Numeric** output of the Unbundle By Name function to the **t** input of the Select function.
- l. Connect the **Minimum Value** output of the Unbundle By Name function to the **f** input of the Select function.
- m. Place a Bundle function inside the case structure.
- n. Connect the **s?t:f** output of the Select function to the **element 0** input of the Bundle function.
- o. Place a VBAI SDK Get Result VI inside the case structure.



- p. Connect the **Maximum Value Result** output of the Unbundle By Name function to the **Measure ID** input of the VBAI SDK Get Result VI.
  - q. Place an Unbundle By Name function inside the case structure.
  - r. Connect the **Result Value** output of the VBAI SDK Get Result VI to the **input cluster** input of the Unbundle By Name function.
  - s. Place a Select function inside the case structure.
  - t. Connect the **Result Available** output of the VBAI SDK Get Result VI to the **s** input of the Select function.
  - u. Connect the **Numeric** output of the Unbundle By Name function to the **t** input of the Select function.
  - v. Connect the **Maximum Value** output of the Unbundle By Name function to the **f** input of the Select function.
  - w. Connect the **s?t:f** output of the Select function to the **element 1** input of the Bundle function.
  - x. Connect the **output cluster** output of the Bundle function to the **Range** input of the IMAQ Threshold VI.
7. Select **File»Save** to save the User Programming VI.
  8. Select **File»Close** to close the User Programming VI.

The User Programming VI uses the VBAI SDK Get Result VI to verify that the GUID for the measurement specified by the user contains a valid numeric value. If the GUID contains a valid numeric value, the User Programming VI performs the image threshold using the measurement value stored in the GUID. If the GUID does not contain a valid numeric value, the User Programming VI uses the value of the corresponding constant.

# Debugging the Custom Step

---

Complete the following steps to debug a custom step:

1. Launch LabVIEW.



**Note** You must configure LabVIEW before debugging a custom step. Refer to the [Configuring LabVIEW to Debug Custom Steps](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for more information about configuring LabVIEW.

2. Select **Tools»<Vision Builder AI>>Test Your Custom Step in Vision Builder AI** to launch Vision Builder AI and test the step.



**Note** You must open Vision Builder AI from LabVIEW to test a custom step that has not been saved for distribution. Refer to the [Distributing The Custom Step](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for information about saving custom steps for use with the Vision Builder AI executable.

3. Open the Vision Builder AI Configuration interface.
4. Add a **Simulate Acquisition** step to the acquisition. Because the Manual Threshold Previous Measurements custom step operates on an image, you must acquire an image to test the custom step.
5. Add a **Calculator** step to the acquisition. You will use result measurements from the Calculator step to test whether the Manual Threshold Previous Measurements custom step correctly handles previous measurements. Complete the following steps to configure the Calculator step:
  - a. The calculator setup wizard launches when you add a Calculator step to the inspection. In the calculator setup wizard, click **Next** twice so that the Output Results list appears.
  - b. Click **Add New Output Result**.
  - c. Enter `Minimum Value` as the **Name** for the output result.
  - d. Click **Add New Output Result**.
  - e. Enter `Maximum Value` as the **Name** for the output result.
  - f. Click **Finish**.
  - g. Click **Show Functions Palette**.
  - h. Connect a Boolean constant to the Step Result indicator.
  - i. Connect a numeric constant to the Minimum Value indicator. Enter 30 as the value for the numeric constant.
  - j. Connect a numeric constant to the Maximum Value indicator. Enter 120 as the value for the numeric constant.

6. Add the Manual Threshold Previous Results custom step to the inspection. By default, custom steps are located on the **Use Additional Tools** palette.

If you did not modify the custom step icon, the custom step displays the default custom step icon.



7. When you open the custom step, Vision Builder AI operates in debugging mode. The front panel of the User Interface VI for the custom step opens in LabVIEW. The front panel of the User Interface VI may open behind the Vision Builder AI window. Open the User Interface VI by clicking the VI window in the taskbar.

8. Select the **Settings** tab.

9. Select **Calculator 1 - Minimum Value** from the Minimum Value Source list.

10. Select **Calculator 1 - Maximum Value** from the Maximum Value Source list.

The main window displays the output image from the Manual Threshold Previous Measurements custom step, which reflects the threshold range specified by the previous measurements from the Calculator step.

11. Click **OK**.

12. Double-click the Calculator step in the inspection to edit it.

13. Enter 50 as the value of the numeric constant connected to the Minimum Value indicator.

14. Enter 100 as the value of the numeric constant connected to the Maximum Value indicator.

15. Click **OK**.

16. Select the Manual Threshold Previous Measurements step in the inspection.

The main window displays the output image from the Manual Threshold Previous Measurements custom step, which reflects the threshold range specified by the modified previous measurements from the Calculator step.

17. When you are satisfied with the performance of the step, save the step for distribution.

Refer to the [Distributing The Custom Step](#) section of Chapter 3, [Creating a Custom Image Processing Step](#), for more information about saving custom steps for distribution.

# Controls and Indicators Used in Source Code VIs

This section contains detailed descriptions of the controls and indicators that appear in the source code VIs of a custom step.

## Init Globals VI Parameters

Table A-1 lists the parameters available for the Init Globals VI.

**Table A-1.** Init Globals VI Elements

Element	Description
<b>Local Name</b>	Name of the step as it appears in Vision Builder AI.
<b>Description</b>	Descriptive text that appears next to the custom step in the Vision Builder AI tools palette.
<b>Version</b>	String to help keep track of the custom step version. To check the version, the developer must read the variable.
<b>Tab Location</b>	Tab of the Vision Builder AI Inspection Tools palette where the custom step is located.
<b>Region</b>	Region where the user interface for the custom step appears.
<b>Use in Product Select</b>	Specifies whether the custom step is available in the Product Select state.
<b>Context Help</b>	File path to a custom compiled HTML help file. Refer to Appendix B, <i>Creating Documentation for the Custom Step</i> , for information about creating help for the custom step.
<b>ROI Tools and Image Support</b>	Available ROI tools. A green LED indicates that the tool appears in the toolbar. A red LED indicates that the tool does not appear in the toolbar.
<b>Localized ROI Tool Name</b>	Display name for each of the available ROI tools. This element is useful if you need to localize ROI tool names.
<b>Default Tool</b>	Default ROI tool for the custom step.

**Table A-1. Init Globals VI Elements (Continued)**

Element	Description
<b>Show Dynamic ROIs</b>	Show/hide previously created ROIs in the Region of Interest control on the User Interface VI front panel.
<b>Supported Image Types</b>	<p>Image types supported by the custom step. A green LED indicates that the image type is supported by the step. A red LED indicates that the image type is not supported. An error will occur if the user attempts to apply the step to an unsupported image type.</p> <p>The Supported Image Types control does not appear in the Simulate Acquisition Step.</p>
<b>Image Producer Step</b>	Enable this control for the custom step to log the modified image and make it available in the <b>Select Image</b> step.



**Note** When you modify the controls and indicators in the Init Globals VI, you must make the new values the default for the control or indicator and save the changes. To make the current values the default for a control or indicator, select **Edit»Make Current Values Default**, and save the VI.

# User Interface VI Parameters

Table A-2 lists the controls available for the Init Globals VI.

**Table A-2.** User Interface VI Controls

Controls	Description
<b>Name in</b>	Name of the step.
<b>Process ID</b>	Unique ID provided from the Vision Builder AI engine.
<b>Image</b>	Processed image from the previous step in the script.
<b>Setup in</b>	Variant that contains the step parameters that were entered during editing and saved when the user clicked OK and inserted the step into the script. Use the Parameters control to save a type definition of the parameters. Use the Variant To Data VI to access data stored in <b>Setup in</b> . Bundle multiple parameters in a cluster and use the To Variant VI to package the cluster for the <b>Setup out</b> indicator.
<b>Edit Mode</b>	FALSE indicates a new instance of the custom step. TRUE indicates that the step is being edited by the user. For example, if a user double-clicks a step that is already in the script, <b>Edit Mode</b> is TRUE.
<b>ROI Event</b>	<p>Notifies you when a region of interest is drawn in the main image window. The <b>ROI Event</b> is fired by Vision Builder AI when you draw an ROI in the main image. The <b>ROI Event</b> returns the following elements:</p> <ul style="list-style-type: none"> <li>The ROI tool used</li> <li>The event type (click, draw, or double-click)</li> <li>The ROI descriptor</li> </ul> <p>To handle the <b>ROI Event</b>, add code to the &lt;ROI Event&gt;: User Event case of the event structure in the User Interface VI.</p>
<b>Event</b>	<p>Refers to activity in the Vision Builder AI interface. This <b>Event</b> is fired by Vision Builder AI to notify you of an action in the Vision Builder AI interface. The <b>Event</b> string contains the type of event.</p> <p>Currently, <b>New Image</b> is the only supported type of event. The <b>New Image</b> event allows you to execute the custom step on a new image while you are still configuring the custom step. The <b>New Image</b> event is fired when you click the <b>Run State Once</b> button on the Vision Builder AI toolbar.</p> <p>To handle the <b>Event</b>, add code to the &lt;Event&gt;: User Event case of the event structure in the User Interface VI.</p>
<b>Reference CoordSys Array</b>	List of available coordinate systems. Use this in conjunction with the Coordsys Utilities to manage coordinate systems.

**Table A-2. User Interface VI Controls (Continued)**

Controls	Description
<p><b>Pass/Fail Condition In</b></p>	<p>Pass/fail conditions of the custom step. This array contains the following elements:</p> <p><b>Used</b>—Boolean that indicates if the pass/fail condition is used.</p> <p><b>Value 1, Value 2</b>—Limit conditions. In execution mode, these values are compared against the Pass/Fail Data produced by the User Programming VI, depending on the relational operator function selected.</p> <p><b>Function</b>—The relational operator function. The following list includes valid values:</p> <p><b>Less</b>—Compares the pass/fail data set in the User Programming VI to <b>Value 1</b> to determine if it is less than <b>Value 1</b>.</p> <p><b>Less or Equal</b>—Compares the pass/fail data set in the User Programming VI to <b>Value 1</b> to determine if it is less than or equal to <b>Value 1</b>.</p> <p><b>Greater</b>—Compares the pass/fail data set in the User Programming VI to <b>Value 1</b> to determine if it is greater than <b>Value 1</b>.</p> <p><b>Greater or Equal</b>—Compares the pass/fail data set in the User Programming VI to <b>Value 1</b> to determine if it is greater than or equal to <b>Value 1</b>.</p> <p><b>In Range</b>—Determines if the pass/fail data set in the User Programming VI is in between <b>Value 1</b> and <b>Value 2</b>.</p> <p>You can design the user interface so that the user can specify the pass/fail condition. Use the Decision Maker (Float) VI to compare the function results to the pass/fail criteria.</p> <p>The pass/fail condition is passed to the custom step user interface so that you can display the information in the appropriate controls on the front panel and store new values when the user modifies the pass/fail condition on the user interface.</p> <p>The following templates demonstrate the functionality of the Pass/Fail Conditions controls and indicators:</p> <p>Processing Step that Logs Measurements</p> <p>Global Step Status</p> <p>You can modify other types of custom steps to use the <b>Pass/Fail Condition In</b> controls and <b>Pass/Fail Condition Out</b> indicator. Refer to the <i>Types of Custom Steps</i> section of Chapter 2, <i>Understanding Custom Steps</i>, for information about each custom step template.</p>



**Table A-2.** User Interface VI Controls (Continued)

Controls	Description
<b>Previous Measurements</b>	<p>Measurements made by previous steps. This array of clusters has as many elements as there are steps in the script. Each element of the array describes the measurements logged by the corresponding step.</p> <p>A cluster contains the following elements:</p> <p><b>Step Name</b>—Name of the step.</p> <p><b>Measurement IDs</b>—Array of clusters describing the measurements logged by the step named <b>Step Name</b>.</p> <p><b>Note:</b> Use the Get Result VI to access information about the measurement, such as the value, type, name, and unit. If you need to use the Previous Measurements control in the step, save the Measurement ID in the Setup variant, which you can access from the User Programming VI. Use the Get Result VI to get the value of the Measurement ID when the User Programming VI is executed. For examples on how to use Previous Measurements, examine the Generate a Report template or the Global Step Status template.</p>
<b>ROI Descriptor</b>	<p>The constant ROI to be saved. The array contains the following elements:</p> <p><b>Global Rectangle</b>—Contains the coordinates of the bounding rectangle.</p> <p><b>Contours</b>—Each of the individual shapes that define an ROI.</p>
<b>Cancelled</b>	Boolean that indicates if the user cancelled the step.

Table A-3 lists the indicators that return data from the User Interface VI.

**Table A-3.** User Interface VI Indicators

Indicators	Description
<b>Name out</b>	Name the user entered for the step.
<b>Setup out</b>	Variant that contains the parameters of the step. If the step requires more than one parameter, bundle the parameters in a cluster, and use the To Variant VI to package the parameters in the variant.
<b>Pass/Fail Condition out</b>	Pass/fail conditions of the custom step.
<b>ROI Descriptor Out</b>	The constant ROI to be saved. If a programmable ROI was selected, the ROI will be saved automatically.

# User Programming VI Parameters

---

Table A-4 lists the controls available for the User Programming VI.

**Table A-4.** User Programming VI Controls

Controls	Description
<b>Image</b>	Processed image from the previous step in the inspection. <b>Tip:</b> This control is used only in Execution mode.
<b>Process ID</b>	Unique ID provided from the Vision Builder AI engine.
<b>Setup</b>	Parameters the user sets in the XXXX - User Interface VI. Use the Variant To Data VI to extract the values stored in this variant.
<b>Mode</b>	The mode of the VI.
<b>ROI Descriptor</b>	The constant ROI to be saved. The array contains the following elements: <ul style="list-style-type: none"> <li>• <b>Global Rectangle</b>—Contains the coordinates of the bounding rectangle.</li> <li>• <b>Contours</b>—Each of the individual shapes that define an ROI.</li> </ul>

Table A-5 lists the indicators that return data from the User Programming VI.

**Table A-5.** User Programming VI Indicators

Indicators	Description
<b>Image Out</b>	<p>Processed image to be provided to the next step in the script. If you are not processing the image, you must still connect <b>Image In</b> to <b>Image Out</b> to ensure that the other steps in the script receive the image. The <b>Image Out</b> control is used only in Execution mode.</p>
<b>Measurements</b>	<p>Measurements created by the step, which can be passed to subsequent steps. The <b>Measurements</b> control is used only in Execution mode. This cluster contains the following elements:</p> <ul style="list-style-type: none"> <li>• <b>Value</b>—Flattened string of the <b>Measurements</b> value. You can store measurements only in the following three LabVIEW data types: double, Boolean, and string. If you choose a double, <b>Value</b> must be the flattened string of a double. If you choose <b>Pass Fail</b> or <b>Boolean</b> for the <b>Type</b> element, <b>Value</b> must be the flattened string of a Boolean. If you choose <b>ID</b> or <b>String</b> in the <b>Type</b> element, <b>Value</b> must be the flattened string of a LabVIEW string. Use the <b>Flatten to String VI</b> to flatten your chosen value type to a string.</li> <li>• <b>Type</b>—Data type of the data.</li> <li>• <b>Name</b>—Measurement name.</li> <li>• <b>Unit</b>—Measurement unit.</li> </ul>
<b>Pass/Fail Data</b>	<p>Pass/fail data that the Decision Maker VI uses to make the pass/fail decision for the step. The <b>Pass/Fail Data</b> control is used only in Execution mode. The <b>Pass/Fail Data</b> array must contain the same number of elements as the <b>Pass/Fail Conditions out</b> indicator on the user interface. The Decision Maker VI applies the pass/fail condition information from the user interface to the pass/fail condition information of the XXXX - User Programming VI.</p> <ul style="list-style-type: none"> <li>• <b>Pass/Fail Data</b>—Number that the Decision Maker VI compares to the pass/fail condition information from the user interface. The Vision Builder AI engine compares the <b>Pass/Fail Data</b> number to <b>Value 1</b> and <b>Value 2</b> of the <b>Pass/Fail Condition out</b> cluster of the XXXX - User Interface VI. The engine bases this comparison on the function specified in <b>Pass/Fail Condition out</b>.</li> <li>• <b>Fail String</b>—Strings that are displayed in the script window of the user interface when a step fails.</li> </ul>

Table A-6 describes the measurements indicator in detail:

**Table A-6.** Measurements Indicator

Element	Description/Data Type
<b>Value</b>	Flattened string of a double, Boolean, or string. To create this parameter, use the Flatten to String VI to flatten the data to a string.
<b>Type</b>	<p>One of the following predefined Vision Builder AI types:</p> <ul style="list-style-type: none"> <li>• <b>Pass Fail</b>—Do not use.</li> <li>• <b>X Position (Pix)</b>—Use this double to log an x-coordinate point.</li> <li>• <b>Y Position (Pix)</b>—Use this double to log a y-coordinate point.</li> <li>• <b>X Position (World)</b>—Do not use. Vision Builder AI automatically creates the Real World coordinates of points if the image is calibrated.</li> <li>• <b>Y Position (World)</b>—Do not use. Vision Builder AI automatically creates the Real World coordinates of points if the image is calibrated.</li> <li>• <b>Score</b>—Double</li> <li>• <b>Distance (Pix)</b>—Double</li> <li>• <b>Distance (World)</b>—Double</li> <li>• <b>Angle</b>—Double</li> <li>• <b>Straightness</b>—Double</li> <li>• <b>Radius (Pix)</b>—Double</li> <li>• <b>Radius (World)</b>—Double</li> <li>• <b>Roundness</b>—Double</li> <li>• <b>Average Intensity</b>—Double</li> <li>• <b>Std Dev Intensity</b>—Double</li> <li>• <b>Minimum Intensity</b>—Double</li> <li>• <b>Maximum Intensity</b>—Double</li> <li>• <b># of Holes</b>—Double</li> <li>• <b>Boolean</b>—Boolean</li> <li>• <b>Numeric</b>—Double</li> <li>• <b>String</b>—String</li> <li>• <b>Area (Pix)</b>—Double</li> <li>• <b>Area (World)</b>—Double</li> <li>• <b>Percentage of Area</b>—Double</li> <li>• <b>ID</b>—String</li> <li>• <b># of Matches</b>—Double</li> <li>• <b># of Objects</b>—Double</li> <li>• <b># of Edges</b>—Double</li> <li>• <b>Orientation</b>—Double</li> <li>• <b>Aspect Ratio</b>—Double</li> </ul>
<b>Name</b>	Description of the measurement.
<b>Unit (optional)</b>	Unit of measure for the measurement.

---

# Creating Documentation for the Custom Step

You may want to provide documentation for the custom step so that the users can access information about how to use the step. To add HTML documentation for the custom step, you must create HTML documentation for the custom step, then compile your HTML documentation into a help file and modify the Init Globals VI to call the help file for the custom step.

## Creating HTML Documentation for the Custom Step

---

Use any text editor or HTML editor to create HTML documentation for the custom step. Save the file with an `.html` extension.

Although not required, you can create three HTML files for the custom step to be consistent with the Vision Builder AI online help. These three HTML files include one describing configuration procedures, one describing the controls, and one listing frequently asked questions regarding the custom step.

## Downloading and Installing HTML Workshop

---

Use Microsoft HTML Help Workshop to create a compiled HTML help file that you can hook into the custom step. Adding documentation to the custom step provides a professional appearance for the step, and helps your users use the step correctly.

Parts of HTML Help Workshop are components of the Windows operating system. However, unless you routinely develop HTML documentation, you may not have the HTML Help Workshop components necessary to develop HTML help.

To determine if you have the components, select **Start>Run**, type `hhw.exe`, and click **OK**. If you receive an error indicating the application could not be found, download and install Microsoft HTML Workshop from [msdn.microsoft.com](http://msdn.microsoft.com). If HTML Help Workshop loads, go to the steps in the [Integrating the Custom Step Documentation](#) section.

For information about creating compiled HTML help, refer to *Help for HTML Help*, which is available from the **Help** menu in HTML Help Workshop.

# Integrating the Custom Step Documentation

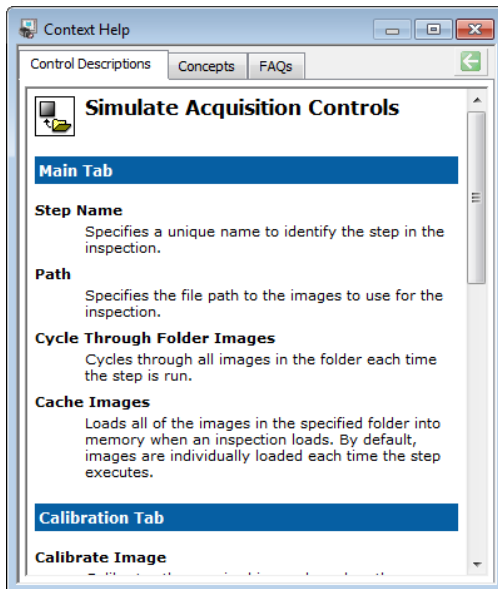
You can add the custom step documentation to the Vision Builder AI user interface so your customers can access the custom step documentation the same way they access the other Vision Builder AI help files.

Complete the following steps to add the custom step documentation to the Vision Builder AI user interface:

1. Save the .chm file to the <Vision Builder AI>\Help folder.
2. Launch LabVIEW.
3. Click **File»Open**. Navigate to <Vision Builder AI>\UserPlugins\  
59 <Custom Step Name>.llb.
4. Select <Custom Step Name> - Init Globals.vi and click **OK**.
5. Type the .chm file name and the HTML page in the elements of the **Context Help** array. For example, if the help file name is myhelp.chm and the HTML file name of the page you that contains configuration information is config.html, enter myhelp.chm\  
config.html for the first element of the array.

Each element of the array is the relative path of the HTML page for the associated tab. For example, the first element is the relative path of the HTML file you want to display for the **Configuration** tab. These tabs appear in the context help window of Vision Builder AI. Refer to Figure B-1 for an example of the context help window.

**Figure B-1.** Context Help Window in Vision Builder AI



If you add at least one file to the **Context Help** array, make sure all three elements of the array are populated, duplicating the file name if necessary. If you add a file name to one element and leave one or more of the remaining elements blank, Vision Builder AI displays an error indicating that a page was not found for the tabs that are represented by the blank elements.

For example, if you created an HTML file that covers configuring the custom step and you add that file name to the first element in **Context Help** but leave the other two elements blank, the **Control Descriptions** and **FAQs** tabs in Vision Builder AI display an error indicating a page was not found for the custom step.

6. Select **Edit»Make Current Values Default**.
7. Save and close the VI.
8. Select **Tools»<Vision Builder AI>»Test Your Custom Step in Vision Builder AI**.
9. Navigate to the appropriate tab and click the custom step to view the newly added documentation.

---

# NI Services

National Instruments provides global services and support as part of our commitment to your success. Take advantage of product services in addition to training and certification programs that meet your needs during each phase of the application life cycle; from planning and development through deployment and ongoing maintenance.

To get started, register your product at [ni.com/myproducts](https://ni.com/myproducts).

As a registered NI product user, you are entitled to the following benefits:

- Access to applicable product services.
- Easier product management with an online account.
- Receive critical part notifications, software updates, and service expirations.

Log in to your National Instruments [ni.com](https://ni.com) User Profile to get personalized access to your services.

---

## Services and Resources

- **Maintenance and Hardware Services**—NI helps you identify your systems' accuracy and reliability requirements and provides warranty, sparing, and calibration services to help you maintain accuracy and minimize downtime over the life of your system. Visit [ni.com/services](https://ni.com/services) for more information.
  - **Warranty and Repair**—All NI hardware features a one-year standard warranty that is extendable up to five years. NI offers repair services performed in a timely manner by highly trained factory technicians using only original parts at a National Instruments service center.
  - **Calibration**—Through regular calibration, you can quantify and improve the measurement performance of an instrument. NI provides state-of-the-art calibration services. If your product supports calibration, you can obtain the calibration certificate for your product at [ni.com/calibration](https://ni.com/calibration).
- **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](https://ni.com/alliance).



- **Training and Certification**—The NI training and certification program is the most effective way to increase application development proficiency and productivity. Visit [ni.com/training](https://ni.com/training) for more information.
  - The Skills Guide assists you in identifying the proficiency requirements of your current application and gives you options for obtaining those skills consistent with your time and budget constraints and personal learning preferences. Visit [ni.com/skills-guide](https://ni.com/skills-guide) to see these custom paths.
  - NI offers courses in several languages and formats including instructor-led classes at facilities worldwide, courses on-site at your facility, and online courses to serve your individual needs.
- **Technical Support**—Support at [ni.com/support](https://ni.com/support) includes the following resources:
  - **Self-Help Technical Resources**—Visit [ni.com/support](https://ni.com/support) 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. Registered users also receive access to the NI Discussion Forums at [ni.com/forums](https://ni.com/forums). NI Applications Engineers make sure every question submitted online receives an answer.
  - **Software Support Service Membership**—The Standard Service Program (SSP) is a renewable one-year subscription included with almost every NI software product, including NI Developer Suite. This program entitles members to direct access to NI Applications Engineers through phone and email for one-to-one technical support, as well as exclusive access to online training modules at [ni.com/self-paced-training](https://ni.com/self-paced-training). NI also offers flexible extended contract options that guarantee your SSP benefits are available without interruption for as long as you need them. Visit [ni.com/ssp](https://ni.com/ssp) for more information.
- **Declaration of Conformity (DoC)**—A DoC is our claim of compliance with the Council of the European Communities using the manufacturer’s declaration of conformity. This system affords the user protection for electromagnetic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting [ni.com/certification](https://ni.com/certification).

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

You also can visit the Worldwide Offices section of [ni.com/niglobal](https://ni.com/niglobal) to access the branch office websites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.