LabVIEW™

Robotics Programming Guide for the *FIRST* Robotics Competition

January 2009
372668D-01

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

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

Worldwide Offices

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

To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

© 2008–2009 National Instruments Corporation. All rights reserved.

Important Information

Warranty

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

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

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

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

Copyright

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

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

Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on `ni.com/legal` for more information about National Instruments trademarks.

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

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`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

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

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

Contents

About This Manual

Conventions xv
Related Documentation xvi
Programming in LabVIEW xvi
Using the CompactRIO Device xvii

Chapter 1
Overview of the *FIRST* Robotics Competition

FRC Terminology 1-1
CompactRIO Device 1-1
Driver Station 1-2
Field Management System 1-2
Autonomous and TeleOp Modes 1-2
Enabled and Disabled Status 1-3
Init, Execute, and Stop Derived States 1-3
Estop State 1-4

Chapter 2
Robot Architecture

CompactRIO Device 2-1
Real-Time Operating System 2-2
FPGA 2-2
I/O Modules 2-2
NI 9201 2-3
NI 9403 2-4
NI 9472 2-4
Axis Camera 2-5
FRC Software LabVIEW Components 2-6
FIRST Robotics Competition VIs 2-6
FRC Configuration Tools 2-6
Setup Axis Camera Tool 2-6
CompactRIO Imaging Tool 2-7

Chapter 3
Configuring the Camera and the CompactRIO Device

Configuring the Axis Camera 3-1
Setting the Static IP Address of the Computer 3-1
Running the Setup Axis Camera Tool 3-3
Configuring the CompactRIO Device 3-3
Setting the Static IP Address of the Computer 3-3
Considerations Before Running the CompactRIO Imaging Tool 3-5
Running the CompactRIO Imaging Tool 3-6
Using the Real-Time System Manager to Manage CompactRIO Device Resources 3-7

Chapter 4
Using the FRC Framework

FRC Robot Project 4-1
Basic FRC Robot Project 4-3
Basic Robot Main VI 4-3
Basic Robot Global VI 4-6
Autonomous Independent VI 4-7
Build DashBoard Data VI 4-7
Dashboard Datatype Type Definition 4-8
Advanced FRC Robot Project 4-8
Robot Main VI 4-9
Team Code VIs 4-10
Type Definitions 4-14
Deploying the FRC Robot Project 4-16
Deploying The Program Using The Run Button 4-17
Deploying The Program From The Project Explorer Window 4-17
Building And Deploying A Stand-Alone Application 4-17
Connecting to the CompactRIO Device 4-19
FRC Dashboard Project 4-19

Chapter 5
Tutorial: Creating an FRC Robot Project

Creating an FRC Robot Project 5-1
Running the FRC Robot Project 5-2
Modifying the Basic Robot Main VI 5-2
Running the FRC Robot Project 5-5
Creating a Stand-Alone FRC Application 5-5

Chapter 6
Tutorial: Creating an FRC Dashboard Project

Displaying Gyroscope Data in the Dashboard Main VI 6-1
Sending Gyroscope Data to the Dashboard Main VI 6-1
Adding an Indicator to the Dashboard Main VI 6-3
Running the Dashboard Main VI 6-6
Displaying Multiple Data Values in the Dashboard Main VI 6-7
Creating a Custom Control 6-7
Sending Multiple Data Values to the Dashboard Main VI 6-8
Adding Indicators to the Dashboard Main VI 6-10
Running the Dashboard Main VI 6-13

Chapter 7
Troubleshooting the FRC Robot

Chapter 8
Using the WPI Robotics Library VIs

Reference Clusters 8-1
Error Handling 8-3

Chapter 9
Accelerometer VIs

Close.vi 9-1
GetAcceleration.vi 9-3
Open.vi 9-6
SetCenterVoltage.vi 9-8
SetGain.vi 9-10

Chapter 10
Accumulator VIs

GetConfiguration.vi 10-1
GetOutput.vi 10-4
Init.vi 10-6
Reset.vi 10-8
SetConfiguration.vi 10-11

Chapter 11
Actuators VIs

Chapter 12
AnalogChannel VIs

Close.vi 12-1
GetAverageVoltage.vi 12-3
GetVoltage.vi 12-5
Open.vi 12-8

Chapter 13
AnalogChannel Advanced VIs

GetAverageValue.vi 13-1
GetAveraging.vi 13-3
GetLSBWeight.vi 13-6
GetOffset.vi 13-8
GetSampleRate.vi 13-11
GetValue.vi 13-12
SetAveraging.vi 13-14
SetSampleRate.vi 13-17

Chapter 14
AnalogTrigger VIs

Close.vi 14-1
GetOutput.vi 14-3
Open.vi 14-5

Chapter 15
Camera VIs

Close.vi 15-1
Get Image.vi 15-3
Get Image From Controller.vi 15-5
Open.vi 15-7
Start.vi 15-9
Stop.vi 15-11

Chapter 16
Camera Properties VIs

Get Brightness.vi 16-1
Get Color Level.vi 16-3
Get Exposure.vi 16-5
Get Exposure Priority.vi 16-7
Get Frame Rate.vi 16-9
Get Image Compression.vi 16-11
Get Image Size.vi 16-13
Get Sharpness.vi 16-15
Get White Balance.vi 16-17
Set Brightness.vi 16-19
Set Color Level.vi 16-21
Set Exposure.vi 16-23
Set Exposure Priority.vi 16-25
Set Frame Rate.vi 16-27
Set Image Compression.vi 16-29
Set Image Size.vi 16-31
Set Sharpness.vi 16-33
Set White Balance.vi 16-35

Chapter 17
Communications VIs

Chapter 18
Compressor VIs

Close.vi 18-1
GetEnableState.vi 18-3
Open.vi 18-6
Start.vi 18-8
Stop.vi 18-10

Chapter 19
Counter VIs

Close.vi 19-2
ConfigureTimer.vi 19-4
Get.vi 19-6
Reset.vi 19-9
Start.vi 19-11
Stop.vi 19-13

Chapter 20
DigitalInput VIs

Close.vi 20-2
GetValue.vi 20-4
Open.vi 20-6

Chapter 21
DigitalOutput VIs

Close.vi 21-2
Open.vi 21-3
SetValue.vi 21-5

Chapter 22
Digital Output Advanced VIs

IsPulsing.vi 22-2
Pulse.vi 22-4

Chapter 23
DriverStation VIs

Get Analog Input.vi 23-1
Get Digital Input.vi 23-2
Get Digital Output.vi 23-2
Set Digital Output.vi 23-3
Set User Data.vi 23-4
Start Communication.vi 23-5
Stop Communication.vi 23-6

Chapter 24
Encoder VIs

Close.vi 24-2
ConfigureTimer.vi 24-4
Get.vi 24-7
Open.vi 24-10
Reset.vi 24-16
Start.vi 24-18
Stop.vi 24-21

Chapter 25
Gyro VIs

Close.vi 25-1
GetAngle.vi 25-3
Open.vi 25-6
Reset.vi 25-8
SetGain.vi 25-10

Chapter 26
I2C VIs

Close.vi 26-1
Open.vi 26-3
Read.vi 26-5
VerifyString.vi 26-7
Write.vi 26-9

Chapter 27
Interrupts VIs

Close.vi 27-1
Open.vi 27-3
Wait.vi 27-5

Chapter 28
IO VIs

Chapter 29
Joystick VIs

Close.vi 29-2
Get.vi 29-2
GetAxis.vi 29-4
GetRawValue.vi 29-5
Open.vi 29-7

Chapter 30
MotorControl VIs

Close.vi 30-1
GetSpeed.vi 30-4
SetSpeed.vi 30-8

Chapter 31
PWM VIs

Close.vi 31-1
GetValue.vi 31-4
Open.vi 31-8
SetPeriodMultiplier.vi 31-11
SetValue.vi 31-14

Chapter 32
Relay VIs

Close.vi 32-3
Open.vi 32-5
Set.vi 32-7

Chapter 33
RobotDrive VIs

Close.vi 33-1
HolonomicDrive.vi 33-4
TankDrive.vi 33-10

Chapter 34
RobotDrive Advanced VIs

Motors.vi 34-1

Chapter 35
Sensors VIs

Chapter 36
Serial Port VIs

Close.vi 36-1
Flush.vi 36-3
GetBytesReceived.vi 36-5
Open.vi 36-7
Read.vi 36-9
Reset.vi 36-11
SetTimeout.vi 36-13
Termination.vi 36-15
Write.vi 36-17

Chapter 37
Servo VIs

Close.vi 37-1
GetAngle.vi 37-4
GetPosition.vi 37-7
Open.vi 37-11
SetAngle.vi 37-14
SetPosition.vi 37-17

Chapter 38
Solenoid VIs

Close.vi 38-3
Get.vi 38-4
Open.vi 38-7
Set.vi 38-9

Chapter 39
SPI VIs

Close.vi 39-1
ConfigureAdvancedOptions.vi 39-3
Open.vi 39-5
Read.vi 39-9
Write.vi 39-11

Chapter 40
Ultrasonic VIs

Close.vi 40-1
GetRange.vi 40-3
Open.vi 40-7
Ping.vi 40-9

Chapter 41
Utilities VIs

FRC FPGAVersion.vi 41-1
FRC LEDs.vi 41-2
FRC ReadSwitch.vi 41-4

Chapter 42
Watchdog VIs

Close.vi 42-1
Delay and Feed.vi 42-3
Feed.vi 42-5
GetStatus.vi 42-7
Kill.vi 42-9
Open.vi 42-10
SetEnabled.vi 42-12

About This Manual

The *FIRST* Robotics Competition (FRC) software includes two separate programming environments, LabVIEW and Wind River Workbench. Use either environment to develop the robotics program you want to run on the CompactRIO device. Use LabVIEW to program a robot in the LabVIEW graphical programming environment. Use Wind River Workbench to program a robot in C or C++.

This manual discusses how to develop a robotics program in LabVIEW. Use this manual to access information about robotics programming concepts and reference information about the *FIRST* Robotics Competition VIs. This manual also describes how to configure the CompactRIO device and the Axis camera, as well as how to use the FRC framework.

Refer to the *C/C++ Programming Guide for the FIRST Robotics Competition*, available by navigating to the `Wind River\docs\extensions\FRC` directory and opening `C Programming Guide for FRC.pdf`, for information about how to develop a robotics program with Wind River Workbench.

Conventions

The following conventions appear in this manual:

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

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

This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.

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

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

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

`monospace bold` Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

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

Related Documentation

The following documents contain information that you may find helpful as you read this manual. Refer to the FRC Community Web site at `www.usfirst.org/community/frc` for official information about the FRC competition, including rules and regulations as well as support information. The *FRC Control System Manual* also is available on this Web site under **Documents and Updates**.

Programming in LabVIEW

The following documents contain information that you may find helpful as you use LabVIEW:

- *Getting Started with LabVIEW for the FIRST Robotics Competition*—Use this manual to learn about the LabVIEW graphical programming environment and the basic LabVIEW features you can use to build FRC applications. Access this manual by navigating to the `National Instruments\LabVIEW 8.5\manuals` directory and opening `FRC_Getting_Started.pdf`.
- *LabVIEW Help*—Use the *LabVIEW Help* to access information about LabVIEW programming concepts, step-by-step instructions for using LabVIEW, and reference information about LabVIEW VIs, functions, palettes, menus, tools, properties, methods, events, dialog boxes, and so on. The *LabVIEW Help* also lists the LabVIEW documentation resources available from National Instruments. Access the *LabVIEW Help* by selecting **Help»Search the LabVIEW Help** in LabVIEW.

- *LabVIEW Quick Reference Card*—Use this card as a reference for information about documentation resources, keyboard shortcuts, data type terminals, and tools for editing, execution, and debugging. Access this manual by navigating to the `National Instruments\LabVIEW 8.5\manuals` directory and opening `LV_Quick_Reference.pdf`.
- *National Instruments FIRST Community*—Refer to the National Instruments *FIRST* Community Web site at `ni.com/first` to access tutorials and examples about using LabVIEW for FRC and to connect with other FRC participants.

Using the CompactRIO Device

The following documents contain information that you may find helpful as you use the CompactRIO device:

- *cRIO-FRC Operating Instructions and Specifications*—Use this manual to learn about installing, configuring, and using the CompactRIO device for the *FIRST* Robotics Competition. Access this manual by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `crio-frc_Operating_Instructions.pdf`.
- *NI 9201/9221 Operating Instructions*—This document describes how to use the National Instruments 9201 and National Instruments 9221 and includes specifications and terminal/pin assignments for the NI 9201/9221. Access this manual by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9201_9221_Operating_Instructions.pdf`.
- *NI 9403 Operating Instructions and Specifications*—This document describes how to use the National Instruments 9403 and includes specifications and pin assignments for the NI 9403. Access this manual by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9403_Operating_Instructions.pdf`.
- *NI 9472/9474 Operating Instructions*—This document describes how to use the National Instruments 9472 and National Instruments 9474 and includes specifications and terminal/pin assignments for the NI 9472/9474. Access this manual by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9472_9474_Operating_Instructions.pdf`.

1

Overview of the *FIRST* Robotics Competition

The objective of the *FIRST* Robotics Competition (FRC) is to build and program a robot to perform certain tasks either autonomously or in response to commands the robot receives from input devices such as joysticks and game controllers.

FRC Terminology

The following terms are useful to know as you build and program a robot for FRC and as you read this manual. Refer to the *FIRST* Community Web site at `www.usfirst.org/community/frc` for more information about the FRC control system and related terminology.

CompactRIO Device

A CompactRIO device, also referred to as a cRIO, is a small, rugged controller consisting of an embedded real-time processor, an FPGA (field-programmable gate array), and slots for interchangeable I/O modules. For FRC, the CompactRIO device contains five I/O modules: two analog input modules, two digital input/output modules, and one digital output module. Each module interfaces directly with the FPGA on the CompactRIO device.

Refer to the *I/O Modules* section of Chapter 2, *Robot Architecture*, for more information about the I/O modules that the FRC kit provides.

The CompactRIO device serves as the “brain” of each robot and runs the program you develop and deploy from LabVIEW. The CompactRIO device also receives data from the driver station and the various sensors that you connect to the I/O modules. Furthermore, the CompactRIO device hosts two watchdogs. The system watchdog shuts down all motors if the communication network fails. The user watchdog can shut down all motors when programming failures, such as infinite loops, occur.

Driver Station

The driver station passes data between the host computer; input devices, such as joysticks and game controllers; field management system (FMS); and the robot. The driver station can read analog data and can both read and write digital data. The driver station receives data from the input devices and sends the data to the CompactRIO device on the robot. The driver station also passes data it receives from the CompactRIO device to the host computer so you can view the data in LabVIEW.

Connect the driver station using an Ethernet connection to a host computer. You can connect the driver station directly to the CompactRIO device or wirelessly through a wireless access point. The wireless access point must be on the same local subnet as the wireless access point you connect to the CompactRIO device.

During the actual FRC competition, connect the driver station to the FMS instead of to a wireless access point. The field management system contains a wireless access point that can communicate with the wireless access points on multiple robots.

Field Management System

FIRST uses an FMS during the FRC competition to monitor robot traffic and to send commands to the driver stations. For example, the FMS designates the current mode of the competition. The FMS can connect through an Ethernet connection to multiple driver stations and through a wireless connection to multiple robots.

Autonomous and TeleOp Modes

The FRC competition consists of two parts: Autonomous mode and TeleOp mode.

In Autonomous mode, the robot moves without receiving commands from input devices, such as joysticks and game controllers. You develop a program in LabVIEW and deploy that program to the CompactRIO device on the robot. When you run the program, the robot moves and behaves according to instructions in the program.

In TeleOp mode, the robot moves in response to commands it receives from input devices. As in Autonomous mode, you develop a program in LabVIEW and deploy that program to the CompactRIO device on the robot. The program must contain instructions that allow the robot to receive

data from the input devices and respond accordingly. When you run the program, you then can use the input devices to manipulate the behavior of the robot. If you want the robot to respond to commands it receives from input devices, connect the input devices to the driver station using a USB connection. You can use the joystick that the FRC kit provides. You also can use any other joystick, game controller, or other input device that you choose.

Enabled and Disabled Status

At certain points in the FRC competition, the program on the CompactRIO device must be running, but the robot cannot move. For example, the program continues running between the Autonomous and Teleop parts of the competition, but no motors can move. During these times, the FMS sets the status of the robot to **Disabled** and kills the system watchdog, which disables the PWM, relay, and solenoid outputs of the robot.

When the robot is in the Disabled status, you still can use input devices, such as joysticks and game controllers, to send information to the robot. You also still can read information from sensors and perform image processing on the CompactRIO device. Therefore, you can develop your robotics program to handle the Disabled status such that the program handles initialization and processing tasks, rather than motion tasks, when the robot is disabled. When the FMS sets the status of the robots to **Enabled**, the system watchdog re-enables the PWM, relay, and solenoid outputs. Therefore, you can develop your robotics program to move motors and drive the robot when the robot is enabled.

Ensure the program you run on the CompactRIO device handles the Disabled and Enabled statuses appropriately. Refer to Chapter 4, *Using the FRC Framework*, for more information about the FRC robot project.

Init, Execute, and Stop Derived States

During the competition, the robot can be in one of the following competition states:

- Autonomous Disabled
- Autonomous Enabled
- TeleOp Disabled
- TeleOp Enabled

For each of these states, the robot can be in one of three derived states: Init, Execute, or Stop. In the Init derived state, you can specify the robot to perform any initialization tasks, such as opening sensor or I/O references. In the Execute derived state, you can specify the robot to move, read and write data, or perform some other behavior. In the Stop derived state, you can specify the robot to perform tasks such as stopping motors and closing references.

Each time the competition state changes, the robot program transitions first to the Stop derived state of the current state and then to the Init derived state of the new state. For example, if the robot is in the Execute derived state of the Autonomous Disabled state and the FMS sets the competition state to **TeleOp Disabled**, the robot transitions first to the Stop derived state of the Autonomous Disabled state and then to the Init derived state of the TeleOp Disabled state.

Estop State

The emergency stop, or Estop, state is an emergency or disqualification state that shuts down the robot immediately. Only the FMS can set the robot to this state. If the robot reaches the Estop state, you must reboot the CompactRIO device to restart execution.

2

Robot Architecture

The *FIRST* Robotics Competition (FRC) controller system consists of the following subsystems:

- **Driver Console**—Establishes communication between the CompactRIO device on the robot and the host computer. This subsystem includes the driver station; one or more input devices, such as joysticks and game controllers; and a host computer.
- **Wireless Communication**—Establishes wireless communication between the driver station and the robot. This subsystem includes wireless access points for the driver station and the CompactRIO device.
- **Robot**—Consists of all parts that make up the moving robot. This subsystem includes the CompactRIO device with five I/O modules, analog and pneumatic breakouts, a digital sidecar, and an Axis camera.

This chapter discusses how to use LabVIEW to interface with the robot subsystem. Refer to the *FRC Control System Manual*, accessible on the FRC Community Web site at `www.usfirst.org/community/frc` under **Documents and Updates**, for more information about the entire FRC controller system.

CompactRIO Device

A CompactRIO device, also referred to as a cRIO, is a small, rugged controller consisting of an embedded real-time processor, an FPGA (field-programmable gate array), and slots for interchangeable I/O modules. For FRC, the CompactRIO device contains five I/O modules: two analog input modules, two digital input/output modules, and one digital output module. Each module interfaces directly with the FPGA on the CompactRIO device.

Real-Time Operating System

Most LabVIEW applications run on a general-purpose operating system (OS) like Windows, Mac OS, or Linux. Some applications require real-time performance that general-purpose operating systems cannot guarantee.

Real-time systems are deterministic. Determinism is the characteristic of a system that describes how consistently the system responds to external events or performs operations within a given time limit. Jitter is a measure of the extent to which execution timing fails to meet deterministic expectations. Most real-time applications require timing behavior to consistently execute within a small window of acceptable jitter.

You can run real-time programs on a real-time target such as a CompactRIO device. The CompactRIO device runs the real-time operating system (RTOS) of Wind River VxWorks. In LabVIEW, deploy each application you want to run on the CompactRIO device from the **RT CompactRIO Target** directory of the **Project Explorer** window. Refer to the *Deploying the FRC Robot Project* section of Chapter 4, *Using the FRC Framework*, for more information about deploying an application to the CompactRIO device.

FPGA

The CompactRIO device includes a built-in, high-performance FPGA. An FPGA is an embedded chip that you can reconfigure for different applications. Each I/O module on the CompactRIO device interfaces directly with the FPGA.

Use the FRC FPGAVersion VI to determine the version and revision of the FPGA on the CompactRIO device. The version of the FPGA corresponds to the year of the FRC competition.

I/O Modules

For FRC, the CompactRIO device contains five I/O modules:

- 2 — NI 9201 analog input
- 2 — NI 9403 digital input/output
- 1 — NI 9472 digital output

Table 2-1 lists the slots on the CompactRIO device and the modules corresponding to those slots.

Table 2-1. Slots for I/O Modules on the CompactRIO Device

Slot	I/O Module
1	NI 9201
2	NI 9201
3	—
4	NI 9403
5	—
6	NI 9403
7	—
8	NI 9472

When you use the WPI Robotics Library VIs, you must specify the module and channel you want to use. For example, you can specify a module value of 1 to use the NI 9201 in slot 1 of the CompactRIO device. You then can select a value for the channel on the NI 9201 that you want to use.

NI 9201

The NI 9201 provides connections for eight analog input channels. You connect an analog breakout to each of the NI 9201 modules. You then can connect any analog sensors, such as accelerometers and gyros, to the analog breakouts to acquire analog data.

Note You must insert the NI 9201 analog modules before the CompactRIO device boots for accurate calibration of the modules. Swapping the modules after the CompactRIO device boots applies the calibration for the original module to the replacement module. Inserting the modules after the CompactRIO device boots applies factory default calibration values as the analog module constants.

Use the Accelerometer VIs and the Gyro VIs to acquire analog data from those sensors. Refer to Chapter 9, *Accelerometer VIs*, and Chapter 25, *Gyro VIs*, for more information about the Accelerometer and Gyro VIs, respectively.

Use the AnalogChannel VIs to acquire analog data from other analog input devices. Refer to Chapter 12, *AnalogChannel VIs*, for more information about the AnalogChannel VIs.

Refer to the *NI 9201/9221 Operating Instructions* document, accessible by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9201_9221_Operating_Instructions.pdf`, for more information about the NI 9201.

NI 9403

The NI 9403 provides connections for 32 digital input or digital output channels. You use a DSUB cable to connect a digital sidecar to each of the NI 9403 modules. You then can connect digital sensors, such as counters, encoders, and ultrasonic sensors, to the digital sidecars to acquire digital data.

Use the Counter, Encoder, and Ultrasonic VIs to acquire digital data from those sensors. Refer to Chapter 19, *Counter VIs*, Chapter 24, *Encoder VIs*, Chapter 26, *I2C VIs*, and Chapter 40, *Ultrasonic VIs*, for more information about these VIs, respectively.

Use the DigitalInput VIs and the DigitalOutput VIs to send or receive digital data from other digital input and output devices. Refer to Chapter 20, *DigitalInput VIs*, and Chapter 21, *DigitalOutput VIs*, for more information about the DigitalInput and DigitalOutput VIs, respectively.

Refer to the *NI 9403 Operating Instructions and Specifications* document, accessible by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9403_Operating_Instructions.pdf`, for more information about the NI 9403.

NI 9472

The NI 9472 provides connections for eight digital output channels. Whereas the NI 9403 can output a maximum voltage of 5.2 V, the NI 9472 can output a maximum voltage of 30 V. If you use the pneumatic breakout with the NI 9472, the NI 9472 can output a maximum voltage of 12 V.

You can use this module to control a solenoid. Connect a pneumatic breakout to the NI 9472 and connect the solenoid to the pneumatic breakout. Then use the Solenoid VIs to specify whether the NI 9472 sends a 12 V signal to the solenoid.

Refer to Chapter 38, *Solenoid VIs*, for more information about the Solenoid VIs.

Refer to the *NI 9472/9474 Operating Instructions* document, accessible by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9472_9474_Operating_Instructions.pdf`, for more information about the NI 9472.

Axis Camera

The robot subsystem includes an Axis camera that you can use to perform image acquisition. You can acquire images and process them directly on the CompactRIO device.

Use the Camera VIs to specify settings for the Axis camera and to acquire images with the camera. Use the *FIRST* Vision VIs to process the images you acquire.

Refer to Chapter 15, *Camera VIs*, for more information about the Camera VIs. Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for more information about the *FIRST* Vision VIs.

The CompactRIO device is headless, so you cannot view the images being acquired. However, as you develop the program you want to run on the CompactRIO device, you can send the image data from the CompactRIO device to the host computer and view the images on the host computer. Use the Get Image From Controller VI to retrieve a specific image on the host computer from the image data that the CompactRIO device sends.

Refer to Chapter 15, *Camera VIs*, for more information about the Get Image From Controller VI.

You also can use live front panel debugging to view the images on the host computer. Click the **Run** button of the VI you want to deploy to the CompactRIO device to perform live front panel debugging.

Refer to the *Deploying The Program Using The Run Button* section of Chapter 4, *Using the FRC Framework*, for more information about live front panel debugging.

Note You can send image data to the host computer or perform live front panel debugging only during development, not during the FRC competition.

Before you can use the Axis camera, you must configure it with the correct username and password. Use the Setup Axis Camera Tool, available by selecting **Tools»Setup Axis Camera**, to configure the Axis camera. Refer to the *Configuring the Axis Camera* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the camera.

FRC Software LabVIEW Components

The FRC software includes LabVIEW, the LabVIEW Real-Time Module, the NI Vision Development Module, NI-RIO, as well as FRC-specific VIs. Use this software to interface directly with the various I/O modules on the CompactRIO device and manipulate the behavior of the robot you build. You can use the FRC software to develop a program that runs on the host computer or that you can deploy and run on the CompactRIO device.

FIRST Robotics Competition VIs

The *FIRST* Robotics Competition VIs are divided into two palettes, ***FIRST* Vision** and **WPI Robotics Library**. Use the *FIRST* Vision VIs to process images you acquire with the Axis camera. Use the WPI Robotics Library VIs to interface with the CompactRIO device and perform tasks such as sending and receiving data from sensors and driving motors. The WPI Robotics Library VIs are analogous to the WPI Robotics Library, a library of C/C++ functions developed by Worcester Polytechnic Institute (WPI) for robotics applications.

Refer to Chapters 9 through 42 for more information about the WPI Robotics Library VIs.

FRC Configuration Tools

The FRC software also includes configuration tools you can use to configure the Axis camera and the CompactRIO device.

Setup Axis Camera Tool

Use the Setup Axis Camera Tool, available by selecting **Tools»Setup Axis Camera**, to configure the Axis camera with the correct username and password.

Refer to the *Configuring the Axis Camera* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the Axis camera.

CompactRIO Imaging Tool

Use the CompactRIO Imaging Tool, available by selecting **Tools» CompactRIO Imaging Tool** in LabVIEW, to configure the CompactRIO device with a start-up image. You can use this tool to switch between the LabVIEW and Wind River Workbench programming environments when developing the program you run on the CompactRIO device. You also can restore an image on the CompactRIO device or update the CompactRIO device with a new name or IP address.

Refer to the *Configuring the CompactRIO Device* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the CompactRIO device.

3

Configuring the Camera and the CompactRIO Device

The CompactRIO device in the *FIRST* Robotics Competition (FRC) kit contains an initial image that allows you to steer a robot with a joystick. You can use this image to verify that the motors, motor controllers, and joystick work as expected. After you verify this behavior, you must configure the Axis camera and the CompactRIO device for use in the FRC competition.

Configuring the Axis Camera

When you configure the Axis camera, you set the username and password of the camera to `FRC` so the Camera VIs can communicate with the camera. You need to configure the camera only once. You do not need to reconfigure the Axis camera during the FRC competition.

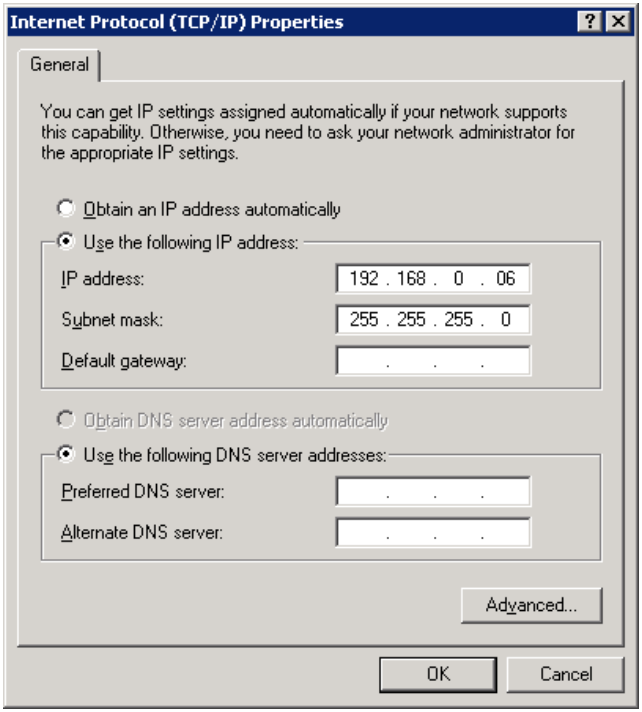
To configure the Axis camera, you must set the static IP address of the computer, connect the computer to the camera, and then run the Setup Axis Camera Tool.

Setting the Static IP Address of the Computer

Complete the following steps to set the static IP address of the computer.

1. Select **Start»Control Panel»Network Connections»Local Area Connection** to display the **Local Area Connection Status** dialog box.
2. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
3. On the **General** page, select **Internet Protocol (TCP/IP)** from the **This connection uses the following items** list.
4. Click the **Properties** button to display the **Internet Protocol (TCP/IP) Properties** dialog box.
5. Select the **Use the following IP address** option.
6. In the **IP address** text box, enter `192.168.0.06`.

7. The **Subnet mask** text box defaults to `255.255.255.0`. Use this default value.



8. Click the **OK** button twice to close the **Internet Protocol (TCP/IP) Properties** and **Local Area Connection Properties** dialog boxes.
9. Click the **Close** button to close the **Local Area Connection Status** dialog box.

After you set the static IP address of the computer, use an Ethernet crossover cable to connect the computer to the camera. Then configure the camera with the Setup Axis Camera Tool.

Running the Setup Axis Camera Tool

Complete the following steps to configure the camera with the Setup Axis Camera Tool.

1. Select **Start»All Programs»National Instruments»LabVIEW 8.5»Setup Axis Camera** to display the **Setup Axis Camera** dialog box. You also can display this dialog box by selecting **Tools»Setup Axis Camera** in LabVIEW.
2. Wait for the dialog box to finish configuration. The **Setup Axis Camera** dialog box sets the username and password of the camera to `FRC` so the Camera VIs can communicate with the camera. If the configuration is unsuccessful, you might need to verify the camera connection and the IP settings.
3. Click the **Close** button to close the **Setup Axis Camera** dialog box.

After you configure the camera, use an Ethernet crossover cable to connect the camera to Ethernet port 2 of the CompactRIO device. The program you run on the CompactRIO device then can communicate with the camera.

You also can configure the username and password of the Axis camera manually. Refer to the *FRC Control System Manual*, accessible on the FRC Community Web site at `www.usfirst.org/community/frc` under **Documents and Updates**, for more information about configuring the camera manually.

Configuring the CompactRIO Device

To configure the CompactRIO device, you must set the static IP address of the computer, connect the computer to the CompactRIO device, and then run the CompactRIO Imaging Tool.

Setting the Static IP Address of the Computer

Complete the following steps to set the static IP address of the computer.

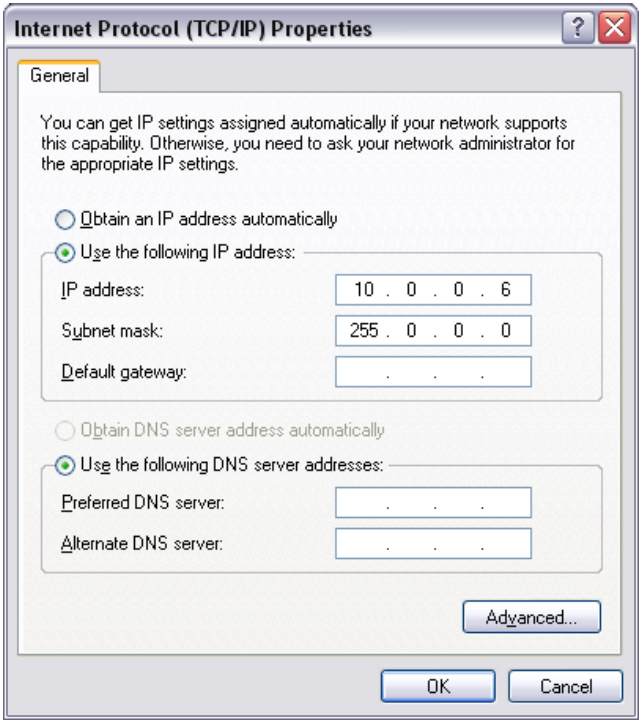
1. Select **Start»Control Panel»Network Connections»Local Area Connection** to display the **Local Area Connection Status** dialog box.
2. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
3. On the **General** page, select **Internet Protocol (TCP/IP)** from the **This connection uses the following items** list.
4. Click the **Properties** button to display the **Internet Protocol (TCP/IP) Properties** dialog box.

5. Select the **Use the following IP address** option.
6. In the **IP address** text box, enter `10.`*`xx`*`.`*`yy`*`.6`, where *yy* corresponds to the last two digits of the team number and *xx* corresponds to the first or first two digits of the team number. Table 3-1 lists examples of the static addresses corresponding to different team numbers.

Table 3-1. Static IP Addresses Corresponding to Team Numbers

Team Number	Static IP Address
64	10.0.64.6
512	10.5.12.6
1024	10.10.24.6

7. The **Subnet mask** text box defaults to **`255.0.0.0`**. Use this default value.



8. Click the **OK** button twice to close the **Internet Protocol (TCP/IP) Properties** and **Local Area Connection Properties** dialog boxes.
9. Click the **Close** button to close the **Local Area Connection Status** dialog box.

After you set the static IP address of the computer, use an Ethernet crossover cable to connect the computer to Ethernet port 1 of the CompactRIO device. Then configure the CompactRIO device with the CompactRIO Imaging Tool.

Considerations Before Running the CompactRIO Imaging Tool

Before configuring the CompactRIO device with the CompactRIO Imaging Tool, you must ensure that the hardware and software are configured properly.

Do not use the CompactRIO Imaging Tool on the CompactRIO device over a wireless connection. If the connection is lost, the data that the CompactRIO Imaging Tool writes to the CompactRIO device is corrupted.

Do not use Measurement and Automation Explorer (MAX) to install additional software on the CompactRIO device. MAX overwrites the FRC VIs on the CompactRIO device, which makes the CompactRIO device unusable for the FRC competition. If you use MAX to install additional software on the CompactRIO device, you must use the CompactRIO Imaging Tool to restore the device to a usable state.

Before running the CompactRIO Imaging Tool, ensure the SAFE MODE switch on the CompactRIO device is turned off. For routine use, do not use the CompactRIO Imaging Tool when the CompactRIO device is in SAFE MODE.

Note Severe corruptions of the software or settings on the CompactRIO device result in the device no longer functioning. If the CompactRIO device is corrupted or if the IP Address is set incorrectly, the device boots only in SAFE MODE. When this occurs, switch the device into SAFE MODE. The CompactRIO Imaging Tool offers to reformat the disk. After the disk has been reformatted, switch the CompactRIO device out of SAFE MODE, reboot, and run the CompactRIO Imaging Tool normally.

Running the CompactRIO Imaging Tool

Complete the following steps to configure the CompactRIO device with the CompactRIO Imaging Tool.

1. Select **Start»All Programs»National Instruments»LabVIEW 8.5»FRC cRIO Imaging Tool** to launch the **CompactRIO Imaging Tool** dialog box. You also can display this dialog box by selecting **Tools»CompactRIO Imaging Tool** in LabVIEW.
2. Select the CompactRIO device you want to configure from the **Select CompactRIO Device** table. This table lists all CompactRIO devices connected to the host computer.
3. In the **Choose Development Environment** section, specify whether you want to run and debug LabVIEW or the C/C++ programs that you run on the CompactRIO device.
4. Place a checkmark in the **Format Controller** checkbox. Use the **Format Controller** section to restore an image on the CompactRIO device or update the CompactRIO device with a new name or team ID.
5. From the **Select Image** list, select the most recent `FRC_2009_xx.zip` file to download the FRC_2009_`xx` image to the CompactRIO device. The FRC_2009_`xx` image consists of both a LabVIEW and a C/C++ program.
6. Enter the name you want to use to identify the CompactRIO device in the **Device name** text box.
7. Enter your team number in the **Team ID** field. The CompactRIO Imaging Tool sets the IP address of the CompactRIO device to `10.`*`xx`*`.`*`yy`*`.2`, where *yy* corresponds to the last two digits of the team number and *xx* corresponds to the first or first two digits of the team number.
8. Click the **Apply** button to apply the changes you made and download the FRC_2009_`xx` image to the CompactRIO device. Do not turn off power to the CompactRIO device or interfere with the network connection while the CompactRIO Imaging Tool downloads the image to the CompactRIO device.
9. Turn the CompactRIO device off and then turn it back on to load the new image.

If you modify the program that you run on the CompactRIO device and want to revert the changes, you can use the CompactRIO Imaging Tool to restore the FRC_2009_`xx` image on the CompactRIO device.

If you modify the program you run on the CompactRIO device and want to switch to the program in the other development environment, select the new development environment from the **Choose Development Environment** section and click the **Apply** button. Switching development environments does not reformat or download a new image to the CompactRIO device.

Using the Real-Time System Manager to Manage CompactRIO Device Resources

You can use the Real-Time System Manager to determine the processor load on the CompactRIO device. You may want to do this if the CompactRIO device is operating slowly. The Real-Time System Manager displays details about VIs running on a real-time (RT) target and provides a dynamic display of the memory and CPU resources for the target. You also can stop VIs and start idle VIs on the RT target using the Real-Time System Manager. From the **Project Explorer** window, select **Tools» Real-Time Module»System Manager** to launch the Real-Time System Manager.

Note When you run the Real-Time System Manager, it takes up a significant amount of the memory on the CompactRIO device.

You also can determine the processor load of the CompactRIO device by using the command `memShow` in the **Terminal** tab in Wind River Workbench or in Microsoft Hyperterminal.

Refer to the *Getting Started with the LabVIEW Real-Time Module* document, available by navigating to the `National Instruments\LabVIEW 8.5\manuals` directory and opening `RT_Getting_Started.pdf`, for more information about the Real-Time System Manager.

Refer to the *Configuring the CompactRIO Device* section of this chapter for more information about configuring the CompactRIO device.

4

Using the FRC Framework

The *FIRST* Robotics Competition (FRC) framework is a set of VIs, organized in LabVIEW projects, that you can use as a template when building a robotics application for FRC. The FRC framework consists of two project templates. Use the FRC robot project template to develop the program you want to deploy to and run on the CompactRIO device. Use the FRC dashboard project template to develop the program with which you want to view data on the host computer.

FRC Robot Project

The FRC robot project starts communication between the CompactRIO device and the driver station, initializes the user watchdog and the Axis camera, and specifies how to handle each competition mode and status of the robot.

Complete the following steps to create an FRC robot project:

1. Click the **FRC cRIO Robot Project** link in the **Getting Started** window to display the **Create New FRC Robot Project** dialog box.
2. In the **Project name** text box, enter the name you want to use to identify the new FRC robot project.
3. In the **Project folder** path, enter the location on the host machine to which you want to save the project files and VIs.
4. In the **cRIO IP address** text box, enter the IP address of the CompactRIO device to which you want to deploy the project. The IP address of the CompactRIO device must be in the form `10.`*`xx`*`.`*`yy`*`.2`, where *yy* corresponds to the last two digits of the team number and *xx* corresponds to the remaining first or first two digits of the team number. You can use the CompactRIO Imaging Tool to set the IP address of the CompactRIO device.
5. Specify whether you want to create the FRC robot project using the **Basic Framework** or the **Advanced Framework**.
6. Click the **Finish** button to close the **Create New FRC Robot Project** dialog box and create the new FRC robot project. LabVIEW displays the new FRC robot project in the **Project Explorer** window.

Figure 4-1 shows a basic FRC robot project template (left) and an advanced FRC robot project template.

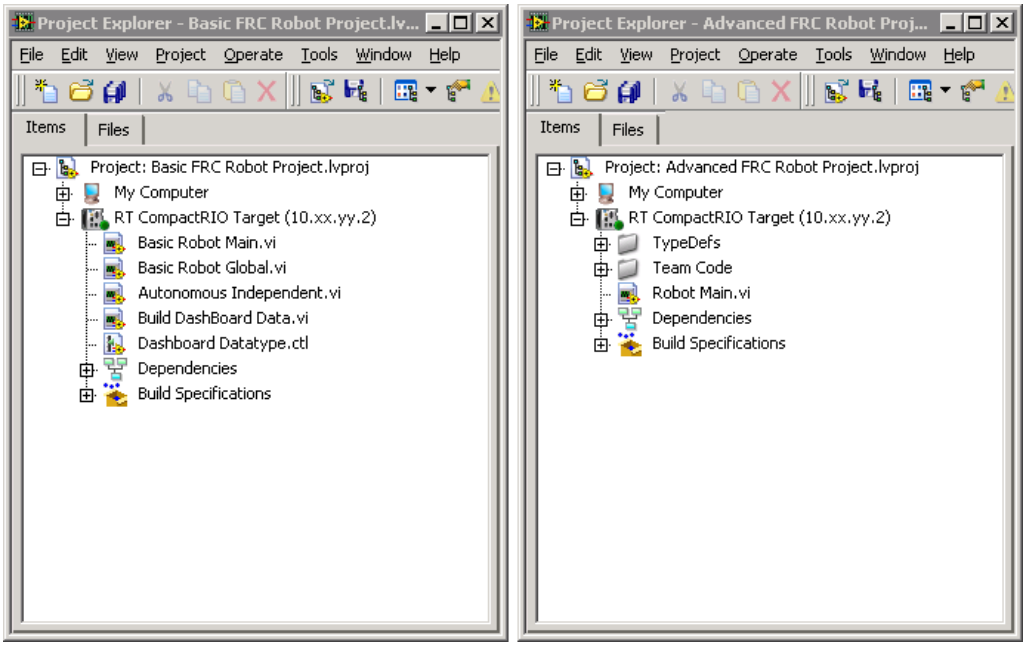


Figure 4-1. FRC Robot Projects

Notice that in each case, the FRC robot project contains two targets: **My Computer** and **RT CompactRIO Target**. Files under **My Computer** are those you want to use and run on the host computer. Files under **RT CompactRIO Target** are those you want to deploy to and run on the CompactRIO device.

Each FRC robot project also has a Robot Main VI. The Robot Main VI is the master VI for the robot and is the top-level VI for the robotics program you run on the CompactRIO device.

Basic FRC Robot Project

If you choose to create a basic FRC robot project from the **Create New FRC Robot Project** dialog box, LabVIEW creates an FRC robot project that looks similar to the following figure:

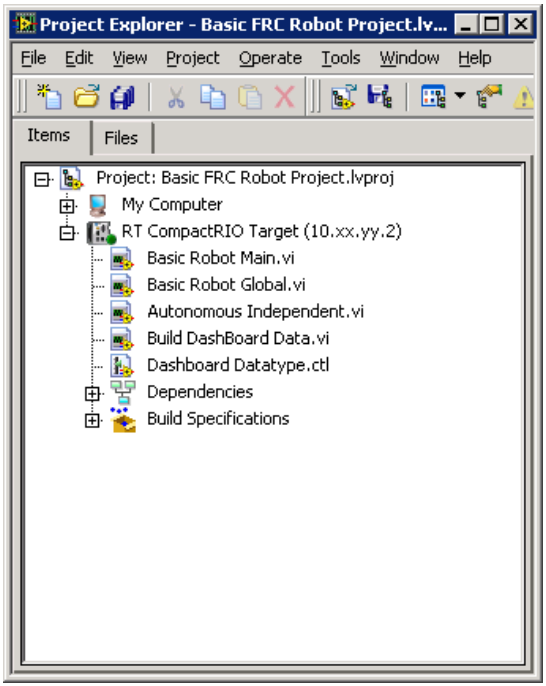


Figure 4-2. Basic FRC Robot Project

The **RT CompactRIO Target** contains four top-level VIs: Basic Robot Main VI, Basic Robot Global VI, Autonomous Independent VI, and Build DashBoard Data VI. The **RT CompactRIO Target** also contains one top-level type definition, the Dashboard Datatype control.

Basic Robot Main VI

The Basic Robot Main VI establishes communication with the driver station, initializes the user watchdog to an enabled state, starts acquiring image data with the Axis camera, and performs Autonomous or TeleOp tasks depending on the competition mode.

In the **Project Explorer** window, double-click the **Basic Robot Main.vi** item to open the Basic Robot Main VI. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. The block diagram contains three While Loops.

Communications While Loop

The first While Loop in the Basic Robot Main VI establishes communication between the CompactRIO device and the driver station. The first While Loop also determines the behavior of the robot according to the competition mode and state. This While Loop contains the Basic Get Mode VI and a Case structure. The Basic Get Mode VI determines the competition state. During the TeleOp portion of the FRC competition, the TeleOp Execute case of the Case structure executes. Use the Autonomous Independent VI to specify the robot behavior for the Autonomous portion of the competition. Refer to the *Autonomous Independent VI* section of this chapter for more information about the Autonomous Independent VI.

By default, the TeleOp Execute case of the Case structure contains the ArcadeDrive VI. With the default configuration, you can use this VI to arcade drive a two-wheeled robot using one joystick and Jaguar motor controllers. Use the WPI Robotics Library VIs and other LabVIEW VIs to modify the TeleOp Execute case of the Case structure to specify the behavior of the robot you build.

The first While Loop also contains the Build DashBoard Data VI, which you can use to send I/O data from the CompactRIO device to the host computer. Refer to the *Build DashBoard Data VI* section of this chapter for more information about the Build DashBoard Data VI.

Notice that the Start Communication VI and the Watchdog Open VI run before the first While Loop. The Start Communication VI starts the communication loop between the CompactRIO device and the driver station and initializes the host computer, joystick, and driver station data caches. This VI runs continuously and regularly checks for information from the host computer, joystick, and driver station. This VI also initializes the system watchdog on the CompactRIO device. The Watchdog Open VI initializes the user watchdog to an enabled state, which ensures that the robot does not continue moving if the program stops executing. For debugging purposes, you might want to disable the user watchdog so the robot can continue moving even if the program reaches a breakpoint. However, be sure the robot is on blocks before running any program with the user watchdog disabled.

When the first While Loop of the Basic Robot Main VI stops, the Stop Communication VI stops the communication loop between the CompactRIO device and the driver station. The Stop Communication VI also releases the host computer, joystick, and driver station data caches.

Image Processing While Loop

The second While Loop of the Basic Robot Main VI processes image data that you acquire from an Axis camera. This While Loop consists of two Case structures. The first Case structure determines whether to start or stop acquiring image data from the Axis camera. The second Case structure determines whether to process the images. If the **Enable Vision** control is set to TRUE, the True case of each Case structure executes. Therefore, the Basic Robot Main VI acquires image data from the Axis camera and processes each image. You can use the *FIRST* Vision VIs to process the image data. If the **Enable Vision** control is set to FALSE, the Basic Robot Main VI neither acquires nor processes any images from the Axis camera.

You might use image processing to help determine the behavior of a robot. For example, you can use the *FIRST* Vision VIs to determine the color of an image you acquire. Depending on the color, you then can move the motors of the robot in an appropriate direction.

If you do not want to perform image acquisition continuously, you can specify to change the value of the **Enable Vision** control depending on the competition mode, for example.

Before the second While Loop runs, the Camera Open, Set Image Size, and Set Frame Rate VIs configure initial settings for the image acquisition. By default, the Basic Robot Main VI configures an image size of 320×240 pixels and a frame rate of 10 frames per second. The IMAQ Create VI specifies a name of **Primary Image** for any image you retrieve from the acquired image data.

When the second While Loop stops, the Camera Stop and Close VIs stop acquiring image data and close the reference to the camera, respectively.

Periodic Tasks While Loop

You can use the third While Loop of the Basic Robot Main VI to perform periodic tasks. For example, you might include PID VIs to perform time-based control operations. By default, this While Loop contains only a Wait (ms) function that waits 100 ms between each iteration of the loop.

Basic Robot Global VI

Use the Basic Robot Global VI to access and pass data among several VIs. In particular, you can use this global VI to store device reference information for the various motors and sensors of your robot.

Because global VIs only pass data and perform no computations, global VIs contain a front panel but no block diagram. By default, the Basic Robot Global VI contains a **RobotDriveDevRef** control and a **WatchdogDevRef** control on the front panel.

On the block diagram of the Basic Robot Main VI, notice that the **RobotDriveDevRef** output of the Open2Motor VI is wired to a global variable. The Open2Motor VI opens a reference to two motors, and the information for these motors is written to the **RobotDriveDevRef** control in the Basic Robot Global VI. If at any time you need to specify a reference to these same motors, you can use the Basic Robot Global VI to provide this data.

Passing Device Reference Data With The Basic Robot Global VI

You can add device references for other motors or sensors to the Basic Robot Global VI in order to pass data for those objects among several VIs. Complete the following steps to use the Basic Robot Global VI to pass data about a gyroscope.

1. Place a Gyro Open VI on the block diagram.
2. Double-click the Gyro Open VI to open the VI.
3. Select the **GyroDevRef** output on the front panel of the Gyro Open VI and press the <Ctrl-C> keys to copy the gyroscope device reference.
4. In the **Project Explorer** window of the basic FRC robot project, double-click the **Basic Robot Global.vi** item to open the Basic Robot Global VI.
5. Press the <Ctrl-V> keys to paste the **GyroDevRef** control on the front panel of the Basic Robot Global VI.
6. Save the Basic Robot Global VI.
7. Drag the VI icon of the Basic Robot Global VI to the block diagram of the VI in which you placed the Gyro Open VI. Notice that LabVIEW places a global variable on the block diagram. By default, the global variable is associated with the first front panel object with an owned label you added to the global VI. In this case, the global variable is associated with the **RobotDriveDevRef** control.

8. Click the **RobotDriveDevRef** global variable and select **GyroDevRef** from the shortcut menu to associate the global variable with the data from the **GyroDevRef** control.
9. Wire the **GyroDevRef** output of the Gyro Open VI to the **GyroDevRef** global variable.
10. When you configure the other parameters of the Gyro Open VI and run the VI, the Gyro Open VI opens a reference to a gyroscope, and LabVIEW writes the gyroscope device reference data to the **GyroDevRef** control of the Basic Robot Global VI.

Autonomous Independent VI

The Autonomous Independent VI is the program you want to run during the Autonomous portion of the FRC competition. Recall that the Basic Robot Main VI references the Autonomous Independent VI and calls it if the competition mode is Autonomous. You do not need to program the Autonomous Independent VI to stop after a certain time because the Basic Robot Main VI terminates this VI when the competition mode changes to TeleOp.

By default, the block diagram of the Autonomous Independent VI contains a For Loop within a Case structure. The Case structure specifies whether to run the default autonomous code for the robot. The False case of the Case structure does nothing. The True case contains a For Loop that moves the robot back and forth slightly four times. You can delete the default autonomous code and use the WPI Robotics Library VIs or other LabVIEW VIs to specify the program you want to run during the Autonomous portion of the FRC competition.

Build DashBoard Data VI

The Build DashBoard Data VI sends I/O data from the modules on the CompactRIO device through the driver station to the host computer. You can use the **Dashboard Enables** input of this VI to specify what data you want to send. By default, the **Dashboard Enables** input specifies to send raw data for the first analog module, the first digital module, and the solenoid module to the host computer. The Build DashBoard Data VI also updates data values in the Dashboard Datatype type definition, which you then can reuse in other VIs.

Dashboard Datatype Type Definition

The Dashboard Datatype type definition specifies data corresponding to the two analog modules, two digital modules, and solenoid module of the CompactRIO device. You can add or remove the controls that this type definition contains depending on the data that you want to reuse.

Advanced FRC Robot Project

The basic FRC robot project has a flat structure that lets you program either Autonomous or TeleOp behavior. However, if you want more control over the robot in each competition state and derived state, you can use the advanced FRC robot project.

If you choose to create an advanced FRC robot project from the **Create New FRC Robot Project** dialog box, LabVIEW creates an FRC robot project that looks similar to the following figure:

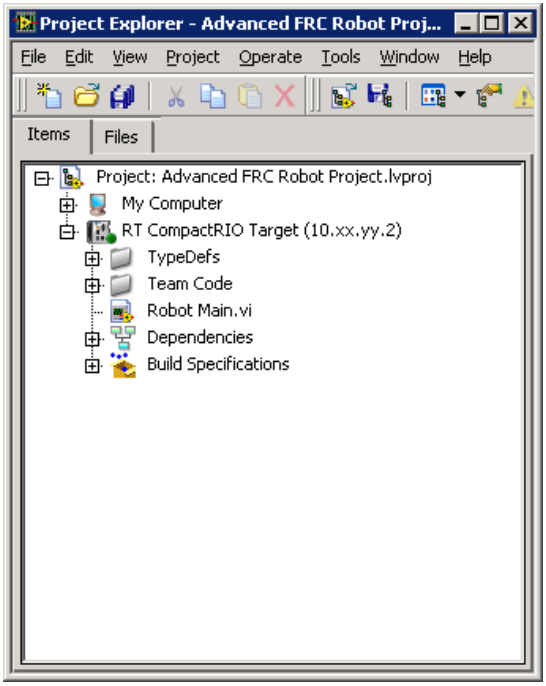


Figure 4-3. Advanced FRC Robot Project

The **RT CompactRIO Target** contains one top-level VI, the Robot Main VI. As with the basic FRC robot project, the Robot Main VI in this project is the top-level VI. This target also contains a **TypeDefs** folder and a **Team Code** folder. These folders contain type definitions and subVIs that the Robot Main VI calls.

Robot Main VI

The Robot Main VI establishes communication with the driver station, acquires and processes images, and performs Autonomous or TeleOp tasks depending on the competition mode.

In the **Project Explorer** window, double-click the **Robot Main.vi** item to open the Robot Main VI. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. The block diagram contains a single While Loop and a number of subVIs. The While Loop determines the behavior of the robot according to the competition mode and state. When the competition state is Autonomous Enabled, the Robot Main VI calls the Autonomous Iterative VI. When the competition state is TeleOp Enabled, the Robot Main VI calls the Teleop VI. If the robot is in Disabled status, the Robot Main VI calls the Disabled VI.

The While Loop also contains the Build DashBoard Data VI, which you can use to send I/O data from the CompactRIO device to the host computer. Refer to the *Build DashBoard Data VI* section of this chapter for more information about the Build DashBoard Data VI.

The other subVIs in the Robot Main VI perform tasks such as initializing data, performing time-based operations, and processing images.

After the Robot Main VI calls the Begin subVI, it uses the Start Communication VI to establish communication with the driver station. The Robot Main VI runs until the **Abort Execution** button on the VI toolbar is pressed or until the **Finish** button on the front panel is pressed. If the **Abort Execution** button is pressed, the Robot Main VI aborts immediately and does not perform any cleanup tasks. If the **Finish** button is pressed, the Robot Main VI calls the Finish VI, which you can configure to close device references, save collected data to file, or perform any other cleanup tasks.

Unlike in the basic FRC robot project, where you modify the Basic Robot Main VI directly, you do not need to modify the Robot Main VI in the advanced FRC robot project. Instead, you only need to modify the code within each of the subVIs in the **Team Code** folder of the FRC robot project.

Team Code VIs

The **Team Code** folder of the FRC robot project contains subVIs that the Robot Main VI calls to perform the initialization, execution, and cleanup tasks for the Autonomous and TeleOp portions of the FRC competition.

Begin VI

The Begin VI initializes data for use throughout the Robot Main VI. You can open references to motors and sensors you want to use, load settings from file, and perform other initialization tasks.

By default, the Begin VI configures settings for the Autonomous program you want to run, opens a reference to the Axis camera, configures options for the FRC dashboard project on the host computer, and defines the RobotData, VisionData, and PeriodicTaskData type definitions. You can use the WPI Robotics Library VIs and other LabVIEW VIs to configure other initialization tasks.

Specify whether you want to use the Autonomous Independent VI or the Autonomous Iterative VI on the block diagram of the Begin VI. If you use the Autonomous Independent VI, you do not need to make any changes from the default values. If you use the Autonomous Iterative VI, you must select **Iterative** from the **Autonomous Style** control and change the **VI Refnum** control from a reference to the Autonomous Independent VI to a reference to the Autonomous Iterative VI.

Autonomous Independent VI

The Autonomous Independent VI is one of two VIs you can choose to run during the Autonomous portion of the FRC competition. This VI runs for the duration of the Autonomous portion of the competition. You do not need to program the Autonomous Independent VI to stop after a certain time because the Robot Main VI terminates this VI when the competition mode changes to TeleOp.

By default, the block diagram of the Autonomous Independent VI contains a For Loop within a Case structure. The Case structure specifies whether to run the default autonomous code for the robot. The False case of the Case structure does nothing. The True case contains a For Loop that moves the robot back and forth slightly four times. You can delete the default autonomous code and use the WPI Robotics Library VIs or other LabVIEW VIs to specify the program you want to run during the Autonomous portion of the FRC competition.

The Autonomous Independent VI is recommended over the Autonomous Iterative VI.

Autonomous Iterative VI

The Autonomous Iterative VI is one of two VIs you can choose to run during the Autonomous portion of the FRC competition. This VI iterates and performs some action each time a packet arrives from the driver station.

The block diagram of the Autonomous Iterative VI consists of two Case structures. The outer Case structure specifies whether to use the Autonomous Independent VI or the Autonomous Iterative VI, as specified on the block diagram of the Begin VI, during the Autonomous portion of the FRC competition. If you choose to use the Autonomous Independent VI, the Autonomous Iterative VI does nothing by default.

If you choose to use the Autonomous Iterative VI during the Autonomous portion of the FRC competition, the inner Case structure executes according to the derived state of the robot. You can use the Init case of the inner Case structure to initialize any motors or sensors for the Autonomous portion of the competition. Alternatively, you can initialize this data in the Begin VI so that the data is available during the TeleOp portion of the competition as well. The Execute case of the inner Case structure performs the action you want to take place each time a packet arrives from the driver station. You can use the Autonomous Elapsed Seconds information, wired to the left border of the Case structure, to determine the action to perform. The Stop case of the inner Case structure performs cleanup tasks, such as closing device references that are needed only during the Autonomous portion of the competition. You can use the Stop case to restore settings to the values they held after the Begin VI ran.

Disabled VI

The Disabled VI runs whenever the robot is in Disabled status. You can use this VI to calibrate sensors or the Axis camera before the FRC competition or between the Autonomous and TeleOp portions of the competition.

The block diagram of the Disabled VI contains a Case structure that executes depending on the derived state of the robot. You can use the Init case of the Case structure to perform initialization tasks that apply only when the robot is in Disabled status. Alternatively, you can initialize this data in the Begin VI so that the data is available when the robot is in Enabled status as well. The Execute case of the Case structure includes any actions, such as calibrating sensors, that you want the robot to perform when the motors are disabled. The Stop case of the Case structure performs

cleanup tasks, such as closing device references that are needed only when the robot is in Disabled status. You can use the Stop case to restore settings to the values they held after the Begin VI ran.

Teleop VI

The Teleop VI iterates and performs some action each time a packet specifying the TeleOp mode arrives from the driver station. This VI only handles the event of the competition mode being set to **TeleOp**. Use the Periodic Tasks VI to program the actual behavior of the robot during the TeleOp portion of the competition.

The block diagram of the Teleop VI contains a Case structure that executes depending on the derived state of the robot. You can use the Init case of the Case structure to initialize any motors or sensors for the TeleOp portion of the competition. Alternatively, you can initialize this data in the Begin VI so that the data is available during the Autonomous portion of the competition as well. By default, the Init case initializes the robot to read the value of joystick 1. The Execute case of the Case structure performs the action you want to take place each time a TeleOp packet arrives from the driver station. For example, you might want to read values from a joystick, update the robot motors, or update setpoints for the periodic operations specified in the Periodic Tasks VI. The Stop case of the Case structure performs cleanup tasks, such as closing device references that are needed only during the TeleOp portion of the competition. You can use the Stop case to restore settings to the values they held after the Begin VI ran.

The Teleop VI uses the RobotData type definition to specify device references of the correct data types for use with the WPI Robotics Library VIs. Refer to the *RobotData Type Definition* section of this chapter for information about the RobotData type definition.

Robot Global Data VI

Use the Robot Global Data VI to access and pass data among several VIs. In particular, you can use this global VI to store device reference information for the various motors and sensors of your robot.

Because global VIs only pass data and perform no computations, global VIs contain a front panel but no block diagram. By default, the Robot Global Data VI contains an **Enable Vision** control and a RobotData type definition on the front panel.

On the block diagram of the Vision Processing VI, the **Enable Vision** global variable is wired to the Case structure that determines whether to start or stop image acquisition. The value of the **Enable Vision** control in the Robot Global Data VI determines which case of the Case structure in the Vision Processing VI to execute.

Vision Processing VI

The Vision Processing VI acquires images from the Axis camera and performs image processing. This VI runs continuously while the Robot Main VI is running.

The block diagram of the Vision Processing VI consists of a While Loop, which itself contains two Case structures. The first Case structure determines whether to start or stop acquiring image data from the Axis camera. The second Case structure determines whether to process the images.

Notice the **Enable Vision** global variable wired to the first Case structure. This global variable is an instance of the Robot Global Data VI. The value of the **Enable Vision** control is set in the Robot Global Data VI and then passed to the Case structure.

If the **Enable Vision** global variable is TRUE, the True case of each Case structure executes. Therefore, the Vision Processing VI acquires image data from the Axis camera, retrieves specific images from this data, and processes each image. You can use the *FIRST* Vision VIs to process the image data. If the **Enable Vision** global variable is FALSE, the Vision Processing VI neither acquires nor processes any images from the Axis camera.

You might use image processing to help determine the behavior of a robot. For example, you can use the *FIRST* Vision VIs to determine the color of an image you acquire. Depending on the color, you then can move the motors of the robot in an appropriate direction.

If you do not want to perform image acquisition continuously, you can configure the **Enable Vision** control in the Robot Global Data VI to change value depending on the competition mode, for example.

Periodic Tasks VI

The Periodic Tasks VI performs periodic tasks. For example, you might include PID VIs to perform time-based control operations. This VI runs continuously while the Robot Main VI is running.

By default, the block diagram of the Periodic Tasks VI consists of two While Loops. Each While Loop contains a Wait (ms) function that specifies the length of time to wait between each iteration of the loop. You can change the value of the **milliseconds to wait** input to specify the frequency of each periodic task.

Periodic loops often operate with setpoints from other loops. Use a global variable such as the Robot Global Data VI to share data among loops.

Build DashBoard Data VI

The Build DashBoard Data VI sends I/O data from the modules on the CompactRIO device through the driver station to the host computer. You can use the **Dashboard Enables** input of this VI to specify what data you want to send. By default, the **Dashboard Enables** input specifies to send raw data for the first analog module, the first digital module, and the solenoid module to the host computer. The Build DashBoard Data VI also updates data values in the Dashboard Datatype type definition, which you then can reuse in other VIs.

Finish VI

The Finish VI performs cleanup tasks before the Robot Main VI stops. This VI runs when you press the **Finish** button on the front panel of the Robot Main VI.

The block diagram of the Finish VI contains a Flat Sequence structure. In the first subdiagram of this structure, you can perform cleanup tasks such as closing device references and saving collected data to file. The second subdiagram of the Flat Sequence structure stops the Finish VI and, in turn, the Robot Main VI.

Type Definitions

The **TypeDefs** folder of the FRC robot project contains type definitions for specifying data types and values for device references that you use with the WPI Robotics Library VIs.

RobotData Type Definition

The RobotData type definition specifies data types and values for the device references you use with the WPI Robotics Library VIs. You can wire elements of this cluster to the appropriate parameters of the WPI Robotics Library VIs. For example, you can wire the **JoystickDevRef** output of the Joystick Open VI to this type definition. The type definition

maintains the values you specified in the Open VI. If you then wire this type definition to the **JoystickDevRef** input of the Joystick Get VI, the Joystick Get VI uses the persisted values of the type definition to determine the joystick for which to return button and axis information.

By default, the RobotData type definition contains WatchdogDevRef, JoystickDevRef, and RobotDriveDevRef device references. Complete the following steps to add a gyroscope device reference to this type definition.

1. In the **Project Explorer** window of the FRC robot project, within the **Team Code** folder under the **RT CompactRIO Target**, double-click the **Begin.vi** item to open the Begin VI.
2. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram.
3. Place a Gyro Open VI on the block diagram near the **Robot Data** indicator.
4. Double-click the Gyro Open VI to open the VI.
5. Select the **GyroDevRef** output on the front panel of the Gyro Open VI and press the <Ctrl-C> keys to copy the gyroscope device reference.
6. On the block diagram of the Begin VI, right-click the **Robot Data** indicator and select **Open Type Def.** from the shortcut menu to open the RobotData type definition. You also can open this type definition from the FRC robot project.
7. Click within the blank area at the bottom of the **RobotData In** cluster.
8. Press the <Ctrl-V> keys to paste the **GyroDevRef** control inside the **RobotData In** cluster. You might need to move the **GyroDevRef** control after pasting it so that it does not overlap with other controls in the cluster. You also might need to extend the bottom border of the **RobotData In** cluster in order to see the entire **GyroDevRef** control.
9. Save the RobotData type definition.
10. On the block diagram of the Begin VI, expand the Bundle By Name function wired to the **Robot Data** indicator so that it displays an additional element.
11. Click the new element of the Bundle By Name function and select **GyroDevRef»All Elements** from the shortcut menu.
12. Wire the **GyroDevRef** output of the Gyro Open VI to the **GyroDevRef** input of the Bundle By Name function.
13. When you configure the other parameters of the Gyro Open VI and run the VI, the Gyro Open VI opens a reference to a gyroscope, and LabVIEW writes the gyroscope device reference data to the **GyroDevRef** control in the RobotData type definition.

VisionData Type Definition

The VisionData type definition specifies data types and values for the device references you use with the WPI Robotics Library VIs in the Vision Processing VI. You can wire elements of this cluster to the appropriate parameters of the WPI Robotics Library VIs.

By default, the VisionData type definition contains a CameraDevRef device reference. You can add additional device references to the type definition in the same way you add device references to the RobotData type definition.

PeriodicTaskData Type Definition

The PeriodicTaskData type definition specifies data types and values for the device references you use with the WPI Robotics Library VIs in the Periodic Tasks VI. You can wire elements of this cluster to the appropriate parameters of the WPI Robotics Library VIs.

By default, the PeriodicTaskData type definition contains WatchdogDevRef, JoystickDevRef, and RobotDriveDevRef device references. You can add additional device references to the type definition in the same way you add device references to the RobotData type definition.

Dashboard Datatype Type Definition

The Dashboard Datatype type definition specifies data corresponding to the two analog modules, two digital modules, and solenoid module of the CompactRIO device. You can add or remove the controls that this type definition contains depending on the data that you want to reuse.

Deploying the FRC Robot Project

After you develop the FRC robot project you want to run, you must deploy the program to the CompactRIO device. You can deploy the program in three ways: using the **Run** button; from the **Project Explorer** window; or as a stand-alone, built application.

Deploying The Program Using The Run Button

Click the **Run** button of the Robot Main VI to deploy the VI to the CompactRIO device. LabVIEW deploys the VI, all items required by the VI, and the target settings to memory on the CompactRIO device.

Note If you click the **Run** button of the Robot Main VI to deploy the VI to the CompactRIO device, the VI might run slowly. Close any subVIs of the Robot Main VI that are open on the host computer to improve performance. Do not close the Robot Main VI.

When you deploy a program with the **Run** button, you maintain a connection between the host computer and the CompactRIO device. The program runs on the CompactRIO device, but you can manipulate the front panel objects of the program from the host computer. You therefore can deploy a program with the **Run** button to perform live front panel debugging.

Note If a program is running on the CompactRIO device and you redeploy that program with the **Run** button, the CompactRIO device stops and restarts the program you deployed. LabVIEW redeploys any VIs that changed or are no longer in memory on the CompactRIO device.

Deploying The Program From The Project Explorer Window

In the **Project Explorer** window, right-click the Robot Main VI and select **Deploy** from the shortcut menu to deploy the VI and any support files for the VI to the target. VIs, libraries, and shared variables are downloaded to memory on the CompactRIO device.

When you deploy a program from the **Project Explorer** window, the program runs only on the CompactRIO device to which it was deployed. Therefore, you cannot perform live front panel debugging.

Building And Deploying A Stand-Alone Application

Build the FRC robot project into a stand-alone application that you then can deploy to the CompactRIO device. You can specify the application to run at startup so the application runs as soon as the CompactRIO is powered on. Deploy stand-alone applications to the CompactRIO device for use in the FRC competition.

Complete the following steps to build the FRC robot project into a stand-alone FRC application and run it on the CompactRIO device at startup.

1. In the **Project Explorer** window, double-click the **FRC Basic Robot Deployment** build specification or the **FRC Robot Boot-up Deployment** build specification under the **Build Specifications** folder to display the **Real-Time Application Properties** dialog box.
2. On the **Information** page, specify a name for the build specification in the **Build specification name** text box.
3. Specify a name for the application in the **Target filename** text box.
4. Specify the location on the host computer to which you want to save the stand-alone application in the **Local destination directory** field.
5. Specify the location on the CompactRIO device to which you want to save the stand-alone application in the **Target destination directory** field.
6. Select **Source Files** from the **Category** list.
7. Verify that the Basic Robot Main VI or the Robot Main VI is in the **Startup VIs** list.
8. Click the **OK** button to close the **Real-Time Application Properties** dialog box.
9. In the **Project Explorer** window, right-click the build specification and select **Build** from the shortcut menu to build the application.
10. Right-click the build specification and select **Run as startup** from the shortcut menu to set the application as the startup application and deploy the application to the CompactRIO device. LabVIEW prompts you to reboot the RT target.
11. Reboot the CompactRIO device to run the application.

Note If you no longer want the application to run on the CompactRIO device at startup, right-click the build specification and select **Unset as startup** from the shortcut menu.

Connecting to the CompactRIO Device

You can connect to the CompactRIO device and access the front panels of VIs in memory on the device. First deploy the VI to the CompactRIO device using the **Run** button, as described in the *Deploying The Program Using The Run Button* section of this chapter. If you stop the VI or close the front panel, the VIs are removed from memory on the CompactRIO device. However, if you only disconnect from the CompactRIO device, the front panel on the host computer appears to stop, but the VI continues running on the CompactRIO device. Right-click the **RT CompactRIO Target** item in the **Project Explorer** window and select **Disconnect** from the shortcut menu to disconnect from the CompactRIO device. If you then close the front panel and reconnect to the CompactRIO device, you re-access the front panels in memory on the device. The front panel of the running VI reappears and displays the current state of the VI. Right-click the **RT CompactRIO Target** item in the **Project Explorer** window and select **Connect** from the shortcut menu to connect to the CompactRIO device.

Note You cannot access the front panels of VIs in memory on a CompactRIO device if a built application is running. You first must stop the running built application or cancel the **Connect** operation.

FRC Dashboard Project

Use the FRC dashboard project on the host computer to view data that the CompactRIO device returns. This project can display images and I/O values that the CompactRIO device sends to the host computer.

Complete the following steps to create an FRC dashboard project.

1. Click the **FRC Dashboard Project** link in the **Getting Started** window to display the **Create New FRC Dashboard Project** dialog box.
2. In the **Project name** text box, enter the name you want to use to identify the new FRC dashboard project.
3. In the **Project folder** path, enter the location on the host machine to which you want to save the project files and VIs.
4. Click the **Finish** button to close the **Create New FRC Dashboard Project** dialog box and create the new FRC dashboard project. LabVIEW displays the new FRC dashboard project in the **Project Explorer** window.

The FRC dashboard project looks similar to the following figure:

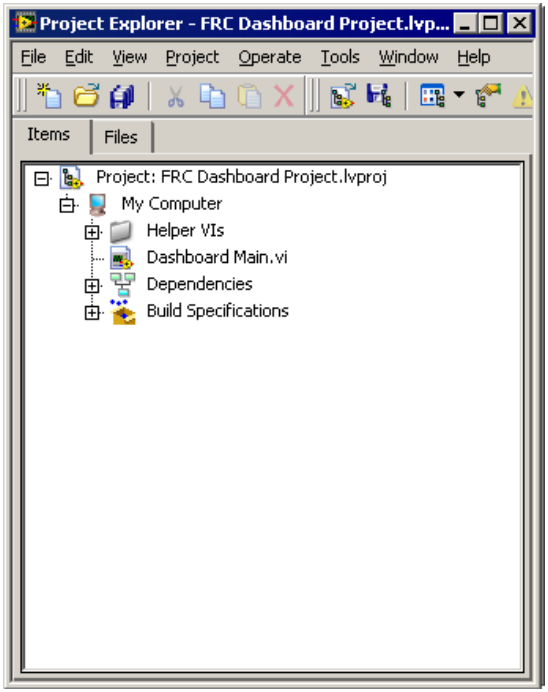


Figure 4-4. FRC Dashboard Project

Whereas the FRC robot project contains two targets, the FRC dashboard project contains only the **My Computer** target. You run the VIs in the FRC dashboard project only on a host computer, such as on a host computer connected to the driver station. You do not deploy these VIs to the CompactRIO device.

The **My Computer** target contains a Dashboard Main top-level VI and a **Helper VIs** folder. The **Helper VIs** folder contains subVIs that the Dashboard Main VI calls. You might not need to modify any of the subVIs in this folder.

The Dashboard Main VI is the master VI in the FRC dashboard project. You can use this VI on the host computer to view image data that the camera connected to the CompactRIO device acquires.

Note You can send image data to the host computer only during development, not during the FRC competition.

You also can use the Dashboard Main VI to read information about the robot, such as error information, robot status, and battery level.

In the **Project Explorer** window, double-click the **Dashboard Main.vi** item to open the Dashboard Main VI. By default, the front panel displays the following information:

- **Image**—Displays the latest image that the camera on the CompactRIO device acquired. The **Video Enable** Boolean control in this section turns image display on or off. Image display might slow performance.
- **Slot 1/Slot 2**—Displays analog input values from the NI 9201 modules in slots 1 and 2 of the CompactRIO device.
- **Slot 4/Slot 6**—Displays digital input or digital output values from the NI 9403 modules in slots 4 and 6 of the CompactRIO device.
- **Slot 8**—Displays digital output values from the NI 9472 module in slot 8 of the CompactRIO device. This module often is used to control a solenoid.
- **Battery Level**—Displays the battery level of the robot.
- **Communications**—Displays input and output values from the driver station.
- **User Data**—Displays user-defined data if you place a checkmark in the **Log** checkbox. Use the Set User Data VI in the FRC robot project to specify the data you want to send from the CompactRIO device to the Dashboard Main VI.
- **Error Messages**—Displays error messages the CompactRIO device sends to the driver station.
- **Match Information**—Displays the competition mode (Autonomous or TeleOp), elapsed time in that mode, and robot status (Enabled or Disabled). The elapsed time starts when the Dashboard Main VI runs and resets when the competition mode changes.
- **Team Logo**—Displays the image saved as `Team Logo.png` or `Team Logo.jpg` in the project directory. You can modify this image to display a logo unique to your FRC team.

Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram of the Dashboard Main VI. The block diagram contains two While Loops.

In the first While Loop, the Dashboard Main VI retrieves specific images on the host computer from the image data that the CompactRIO device sends. By default, the Dashboard Main VI creates a JPEG image called **Host Camera Image**, continuously replaces this image with the most recent image data from the camera on the CompactRIO device, and displays the image in the **Image** indicator on the front panel.

In the second While Loop, the Dashboard Main VI receives data about the robot from the driver station. By default, the Dashboard Main VI connects to the driver station through a UDP connection and acquires status information and I/O data about the robot.

You can use the WPI Robotics Library VIs and other LabVIEW VIs to modify the types of data the Dashboard Main VI displays.

5

Tutorial: Creating an FRC Robot Project

This chapter describes how to create, modify, and deploy a *FIRST* Robotics Competition (FRC) robot project to the CompactRIO device. In this tutorial, you develop a program to perform tank driving with two joysticks and Jaguar motor controllers.

Creating an FRC Robot Project

Complete the following steps to create an FRC robot project that uses the basic framework.

1. Click the **FRC cRIO Robot Project** link in the **Getting Started** window to display the **Create New FRC Robot Project** dialog box.
2. In the **Project name** text box, enter the name you want to use to identify the new FRC robot project.
3. In the **Project folder** path, enter the location on the host machine to which you want to save the project files and VIs.
4. In the **cRIO IP address** text box, enter the IP address of the CompactRIO device to which you want to deploy the project. The IP address of the CompactRIO device must be in the form `10.`*`xx`*`.`*`yy`*`.2`, where *yy* corresponds to the last two digits of the team number and *xx* corresponds to the remaining first or first two digits of the team number.

   You can use the CompactRIO Imaging Tool to set the IP address of the CompactRIO device. Refer to the *Running the CompactRIO Imaging Tool* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the CompactRIO device.
5. Select the **Basic Framework** option to specify that you want to program in the basic framework.
6. Click the **Finish** button to close the **Create New FRC Robot Project** dialog box and create the new FRC robot project.

LabVIEW displays the new FRC robot project in the **Project Explorer** window.

Running the FRC Robot Project

You can deploy the FRC robot project to the CompactRIO device before making any modifications. In the **Project Explorer** window, right-click the **Basic Robot Main.vi** item and select **Run** from the shortcut menu. LabVIEW deploys the Basic Robot Main VI and any support files for the VI to the CompactRIO device. The Basic Robot Main VI then runs on the CompactRIO device. If the robot has a joystick connected to port 1 of the driver station and Jaguar motor controllers controlling the two wheels, you can move the joysticks and observe how the robot responds.

Modifying the Basic Robot Main VI

By default, you can use the Basic Robot Main VI to arcade drive a two-wheeled robot using one joystick and Jaguar motor controllers.

Refer to Chapter 4, *Using the FRC Framework*, for more information about the Basic Robot Main VI.

Complete the following steps to modify the Basic Robot Main VI to perform tank driving with two joysticks and Jaguar motor controllers.

1. Double-click the **Basic Robot Main.vi** item in the **Project Explorer** window to open the Basic Robot Main VI.
2. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram.
3. Click the increment or decrement arrows of the Case structure in the Communications While Loop to select the TeleOp Execute case.
4. If the **Functions** palette is not visible on the block diagram, select **View»Functions Palette**.
5. Place an Open VI from the **Joystick** palette on the block diagram below the existing Joystick Open VI.
6. Right-click the **JoystickDevice** input of the Open VI you placed and select **Create»Constant** from the shortcut menu.
7. From the **JoystickDevice** enum constant, select **USB 2** to use the joystick connected to port 2 of the driver station.

8. Wire the **JoystickDevRef** output of the second Joystick Open VI to the left border of the Communications While Loop to create an input tunnel.

The affected portion of the block diagram should appear similar to the following figure.

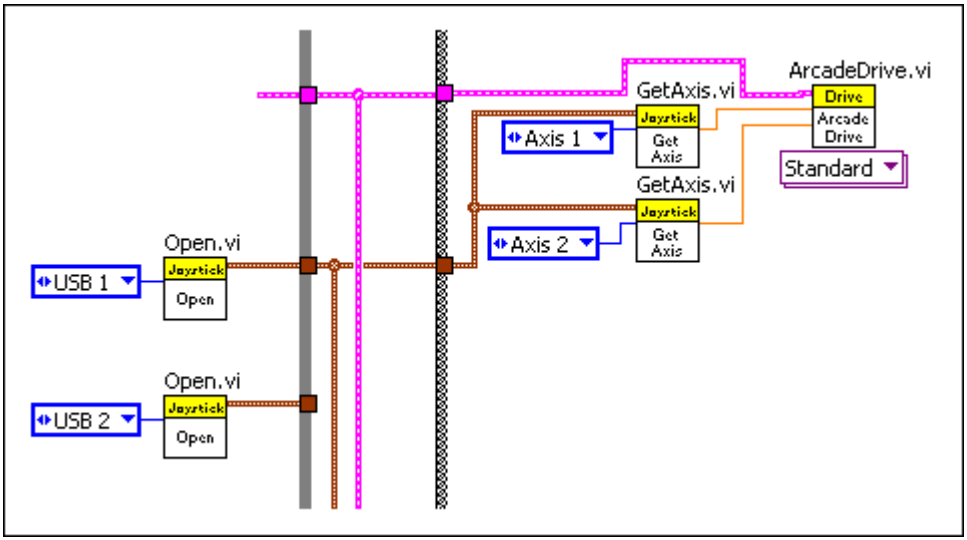


Figure 5-1. Basic Robot Main VI with Additional Joystick Open VI

9. Wire the input tunnel of the While Loop corresponding to the **JoystickDevRef** output of the second Joystick Open VI to the right border of the While Loop to create an output tunnel. Be sure to wire around the Case Structure.
10. Place a Close VI from the **Joystick** palette on the block diagram above the existing Joystick Close VI. You may need to move the existing VIs to make space for the new Joystick Close VI.
11. Wire the output tunnel you created to the **JoystickDevRef** input of the second Joystick Close VI.
12. Delete the wire connecting the first Joystick Open VI to the **JoystickDevRef** input of the second Joystick GetAxis VI.
13. Wire the **JoystickDevRef** output of the second Joystick Open VI to the **JoystickDevRef** input of the second Joystick GetAxis VI.
14. Set the **Axis** enum constant of the first Joystick GetAxis VI to **Axis 2**.

15. Right-click the ArcadeDrive VI and select **Replace»RobotDrive Palette»TankDrive.vi** from the shortcut menu to perform tank driving using the TankDrive VI.

The affected portions of the block diagram should appear similar to the following figure.

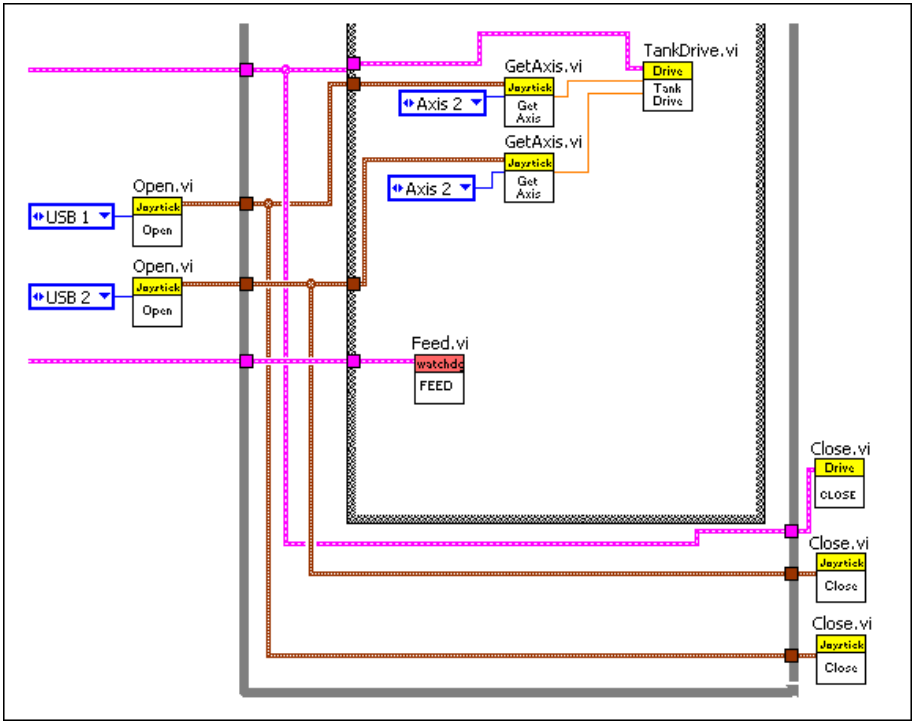


Figure 5-2. Modifying the Basic Robot Main VI for Tank Driving

The Basic Robot Main VI now allows you to perform tank driving. However, this VI cannot communicate with the joysticks or the motor controllers from the host computer. You must deploy the FRC robot project that contains the Basic Robot Main VI to the CompactRIO device. You then can run the Basic Robot Main VI on the CompactRIO device and communicate with the joysticks and motor controllers connected to the device.

Running the FRC Robot Project

You can run the FRC robot project on the CompactRIO device and maintain a connection with the host computer to perform live front panel programming and debugging. By maintaining a connection with the host computer, you can monitor indicators and observe how changes to the front panel of VIs affect the behavior of the robot.

Complete the following steps to run the FRC robot project and perform live front panel debugging.

1. In the **Project Explorer** window, double-click the **Basic Robot Main.vi** item to open the Basic Robot Main VI.
2. Click the **Run** button of the Basic Robot Main VI to deploy the VI to the CompactRIO device. LabVIEW deploys the VI, all items required by the VI, and the target settings to memory on the CompactRIO device.
3. Move the joysticks and observe how the robot responds.
4. Click the **Abort** button of the Basic Robot Main VI. Notice that the VI stops. When you deploy a program with the **Run** button, the program runs on the CompactRIO device, but you can manipulate the front panel objects of the program from the host computer.

Note If you redeploy the Robot Main VI with the **Run** button, the CompactRIO device stops and restarts the Robot Main VI. LabVIEW redeploys any VIs that changed or are no longer in memory on the CompactRIO device.

Refer to the *Deploying the FRC Robot Project* section of Chapter 4, *Using the FRC Framework*, for more information about deploying an FRC Robot Project.

Creating a Stand-Alone FRC Application

You can build the FRC robot project into a stand-alone application that you then can deploy to the CompactRIO device. You can specify the application to run at startup so the application runs as soon as the CompactRIO is powered on. You must deploy stand-alone applications to the CompactRIO device for use in the FRC competition.

Use **Build Specifications** in the **Project Explorer** window to create build specifications for source distributions, such as stand-alone applications. A build specification contains all the settings for the build, such as files to include, directories to create, and settings for VIs.

The default FRC robot project includes the **FRC Basic Robot Deployment** build specification. Use the **Real-Time Application Properties** dialog box to define and modify the settings for a build specification. In the **Project Explorer** window, double-click the **FRC Basic Robot Deployment** build specification under **Build Specifications** to display the **Real-Time Application Properties** dialog box.

Use the **Information** page of the **Real-Time Application Properties** dialog box to specify a name, executable filename, local destination, and target destination for a stand-alone real-time application. You also can compose a description of the build specification. On the Information page, notice that the **Build specification name** is `FRC Basic Robot Deployment`. Notice also the **Local destination directory** path. You might need to change this path to an appropriate location on the host computer to which you want to save the application.

Use the **Source Files** page of the **Real-Time Application Properties** dialog box to specify which VIs automatically run when the application launches and which VIs always are deployed to the CompactRIO device. On the **Source files** page, notice that the Basic Robot Main VI is configured by default to run when the application launches.

When you verify the build specification settings, complete the following steps to build the FRC robot project into a stand-alone FRC application and run it on the CompactRIO device at startup.

1. In the **Project Explorer** window, right-click the **FRC Basic Robot Deployment** build specification under **Build Specifications** and select **Build** from the shortcut menu to build the application.
2. Right-click the build specification and select **Run as startup** from the shortcut menu to set the application as the startup application and deploy the application to the CompactRIO device. LabVIEW prompts you to reboot the real-time target.
3. Reboot the CompactRIO device to run the application.

Note If you no longer want the application to run on the CompactRIO device at startup, right-click the build specification and select **Unset as startup** from the shortcut menu.

4. Move the two joysticks and observe how the motors of the robot respond.

6

Tutorial: Creating an FRC Dashboard Project

This chapter describes how to send data to the Dashboard Main VI and how to modify the *FIRST* Robotics Competition (FRC) dashboard project.

The Dashboard Main VI in the FRC dashboard project displays information about the robot, such as error information, robot status, and battery level. You can use the Dashboard Main VI during robot testing to receive live feedback from the robot. The Dashboard Main VI displays the current input and output values from the CompactRIO device, the robot, and the driver station. This VI also displays user-defined data. Refer to the *FRC Dashboard Project* section of Chapter 4, *Using the FRC Framework*, for more information about the Dashboard Main VI.

The **User Data** section of the Dashboard Main VI displays user-defined messages and information. In this tutorial, you learn how to display gyroscope information from the robot in the **User Data** section of the Dashboard Main VI.

Displaying Gyroscope Data in the Dashboard Main VI

By default, the Dashboard Main VI does not display gyroscope data. You must acquire this data from a gyroscope, send the data to the Dashboard Main VI, and then modify the Dashboard Main VI to display this data.

Sending Gyroscope Data to the Dashboard Main VI

The Gyro Example VI, accessible by clicking the **Gyro Example** link in the **Getting Started** window, acquires information from a gyroscope. You can modify the Gyro Example VI to send the gyroscope information to the Dashboard Main VI.

Complete the following steps to modify the Gyro Example VI to send data to the Dashboard Main VI.

1. Click the **Gyro Example** link in the **Getting Started** window to display the Gyro Example FRC robot project in the **Project Explorer** window.
2. Select **File»Save As** to display the **Save As** dialog box.
3. Select **Copy** and click the **Continue** button to create a copy of the `.lvproj` file on disk.
4. Save the project as `Modified Gyro Example.lvproj` in an easily accessible location.
5. Close the **Project Explorer** window for the Gyro Example FRC robot project. Do not save any changes.
6. Open the Modified Gyro Example project you saved to display the Modified Gyro Example FRC robot project in the **Project Explorer** window.
7. Double-click the **Gyro Example.vi** item in the **Project Explorer** window to open the Gyro Example VI. This VI displays the current angle of the gyroscope you specify.
8. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to display the block diagram.

Note You might need to resize the While Loop and clear some space below the Quotient & Remainder function on the block diagram of the Gyro Example VI before completing the following steps.

9. Place a Number To Exponential String function below the Quotient & Remainder function.
10. Wire the **x-y*floor(x/y)** output of the Quotient & Remainder function to the **number** input of the Number To Exponential String function. The **x-y*floor(x/y)** output corresponds to the gyroscope angle.
11. Place a String To Byte Array function to the right of the Number To Exponential String function.
12. Wire the **E-format string** output of the Number To Exponential String function to the **string** input of the String To Byte Array function.
13. Place a Set User Data VI to the right of the String To Byte Array function. This VI specifies the user data that the Dashboard Main VI receives.
14. Wire the **unsigned byte array** output of the String To Byte Array function to the **User Data (984 bytes)** input of the Set User Data VI.

The affected portion of the block diagram should appear similar to the following figure.

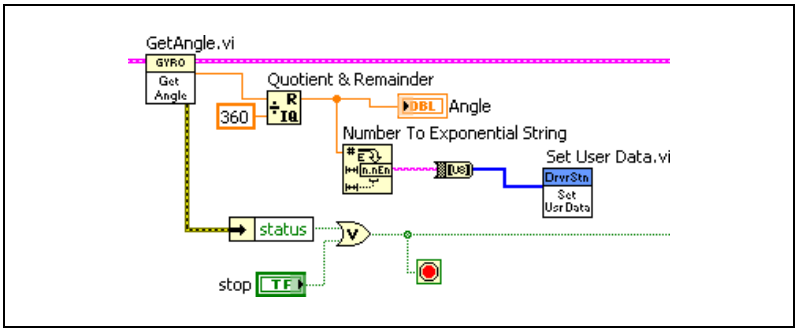


Figure 6-1. Sending Gyroscope Data to the Dashboard Main VI

15. Select **File»Save As** and save the VI as `User Data Gyro Example.vi` in an easily accessible location.

The default Gyro Example VI returns the gyroscope angle as a double-precision, floating-point data type. The modified VI converts this data to an array of bytes, which the Set User Data VI can accept. The Set User Data VI then sends this data to the Dashboard Main VI.

Adding an Indicator to the Dashboard Main VI

The Gyro Example VI now sends the gyroscope data to the Dashboard Main VI. This section describes how to set up the Dashboard Main VI to receive and display the user data. Complete the following steps to display the gyroscope data in the Dashboard Main VI.

1. Select **View»Getting Started Window** to display the **Getting Started** window.
2. Click the **FRC Dashboard Project** link in the **Getting Started** window to display the **Create New FRC Dashboard Project** dialog box.
3. Enter `FRC Dashboard Project User Data Example` in the **Project name** text box of the **Create New FRC Dashboard Project** dialog box.
4. Click the **Finish** button in the **Create New FRC Dashboard Project** dialog box to create a new FRC dashboard project.
5. Double-click the **Dashboard Main.vi** item in the **Project Explorer** window to open the Dashboard Main VI.

6. Place a **Gauge** indicator on the front panel below the **User Data** section of the front panel.
7. Right-click the **Gauge** indicator and select **Properties** from the shortcut menu to display the **Knob Properties** dialog box.
8. On the **Appearance** page, enter `Gyroscope Angle` in the **Label Text** text box to change the name of the indicator.
9. On the **Scale** page, enter `360` as the **Maximum** value in the **Scale Range** section. The **Gyroscope Angle** indicator now displays the heading of the gyroscope within a range of 360 degrees.
10. Click the **OK** button to apply the new value.
11. Hover over the 360 marker on the **Gyroscope Angle** indicator and use the Operating tool to drag the marker to the 0 value. When you drag a marker, the cursor changes to a circular arrow to indicate the tool is over a marker.
12. Right-click the **Gyroscope Angle** indicator and select **Visible Items» Digital Display** from the shortcut menu to add a digital display to the **Gyroscope Angle** indicator.

The affected portion of the front panel should appear similar to the following figure.

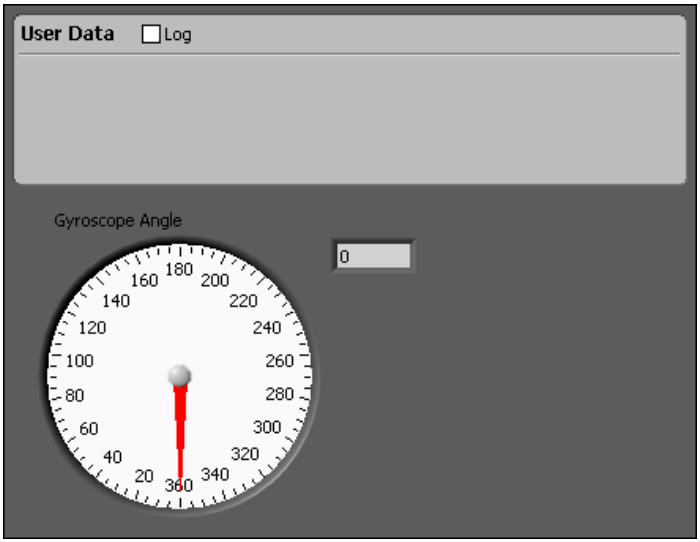


Figure 6-2. Displaying Gyroscope Data with a Gauge Indicator

13. Double-click the **Gyroscope Angle** indicator to display the location of the **Gyroscope Angle** indicator on the block diagram.

Note You might need to resize the While Loop and clear some space to the right of the **Battery level** indicator on the block diagram of the Dashboard Main VI before completing the following steps.

14. Place the **Gyroscope Angle** indicator in the second While Loop of the block diagram to the right of the **Battery level** indicator.
15. Place an Unbundle function between the **Battery level** indicator and the **Gyroscope Angle** indicator.
16. Wire the **Strings** output of the Receive DS Packet VI to the **cluster** input of the Unbundle function. The Receive DS Packet VI receives the gyroscope data that the Gyro Example VI sends.
17. Place a Fract/Exp String To Number function between the Unbundle function and the **Gyroscope Angle** indicator.
18. Wire the **User Data** output of the Unbundle function to the **string** input of the Fract/Exp String To Number function.
19. Wire the **number** output of the Fract/Exp String To Number function to the **Gyroscope Angle** indicator.

The affected portion of the block diagram should appear similar to the following figure.

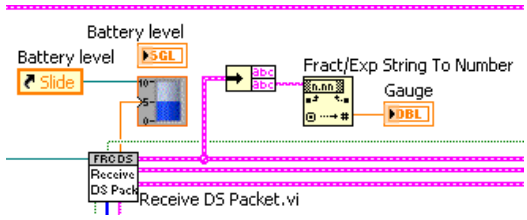


Figure 6-3. Displaying Gyroscope Data in the Dashboard Main VI.

20. Select **File»Save As** and save the VI as `User Data Dashboard Main.vi` in an easily accessible location.

The User Data Dashboard Main VI now receives the gyroscope data from the User Data Gyro Example VI and displays the data in the **Gyroscope Angle** indicator on the front panel.

Running the Dashboard Main VI

Complete the following steps to run the User Data Gyro Example and User Data Dashboard Main VIs.

1. Switch to the User Data Gyro Example VI.
2. Verify that you configured the IP address of the CompactRIO device correctly. Refer to the *Configuring the CompactRIO Device* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the IP address of the CompactRIO device.
3. Click the **Run** button to deploy the User Data Gyro Example VI to the CompactRIO device. The User Data Gyro Example VI runs on the CompactRIO device and sends gyroscope data to the Dashboard Main VI.
4. Switch to the User Data Dashboard Main VI.
5. Verify that you configured the IP address of the host computer correctly. Refer to the *Setting the Static IP Address of the Computer* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the IP address of the host computer.
6. Click the **Run** button to run the User Data Dashboard Main VI on the host computer. This VI now reflects the changes in the angle of the gyroscope.
7. Stop and close all VIs and projects.

This tutorial demonstrated how to send gyroscope data to the Dashboard Main VI. Use the techniques in this tutorial to display any user data in the Dashboard Main VI.

Refer to the *FRC Dashboard Project* section of Chapter 4, *Using the FRC Framework*, for more information about the FRC dashboard project.

Displaying Multiple Data Values in the Dashboard Main VI

In the *Adding an Indicator to the Dashboard Main VI* section of this chapter, you learned how to display gyroscope data in the Dashboard Main VI. In that tutorial, you displayed only one value in the Dashboard Main VI. In this section, you learn how to display multiple values in the Dashboard Main VI using a custom control.

Creating a Custom Control

You can use a custom control to convert multiple data values that the Gyro Example VI sends to the Dashboard Main VI into a data type the Dashboard Main VI can accept. Complete the following steps to create a custom control.

1. Select **File»New** to display the **New** window.
2. Select **Other Files»Custom Control** from the **Create New** list and click the **OK** button to create a custom control. A custom control extends the available set of front panel objects for use in another VI.
3. Select **Strict Type Def.** from the **Control Type** pull-down menu on the toolbar. Specifying the control type as **Strict Type Def.** creates a custom control that reflects any data type changes you make to the custom control in all instances of every VI that uses the control. Also, cosmetic changes you make to a strict type definition affect all instances of the strict type definition.
4. Place a cluster shell on the front panel.
5. Add a numeric control and a string control to the cluster.

The custom control should appear similar to the following figure.

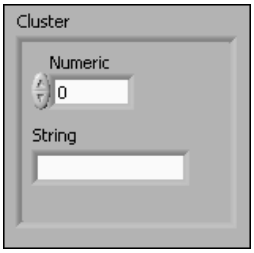


Figure 6-4. Strict Type Definition with Two Data Values

6. Save this control as `myTypeDef.ctl` in an easily accessible location.

Sending Multiple Data Values to the Dashboard Main VI

The Gyro Example VI, accessible by clicking the **Gyro Example** link in the **Getting Started** window, acquires information from a gyroscope. You can modify the Gyro Example VI to send this information to the Dashboard Main VI.

Complete the following steps to modify the Gyro Example VI to send multiple data values to the Dashboard Main VI.

1. Select **View»Getting Started Window** to display the **Getting Started** window.
2. Click the **Gyro Example** link in the **Getting Started** window to display the Gyro Example FRC robot project in the **Project Explorer** window.
3. Select **File»Save As** to display the **Save As** dialog box.
4. Select **Copy** and click the **Continue** button to create a copy of the `.lvproj` file on disk.
5. Save the project as `Multiple Data Gyro Example.lvproj` in an easily accessible location.
6. Close the **Project Explorer** window for the Gyro Example FRC robot project. Do not save any changes.
7. Open the Multiple Data Gyro Example project you saved to display the Multiple Data Gyro Example FRC robot project in the **Project Explorer** window.
8. Double-click the **Gyro Example.vi** item in the **Project Explorer** window to open the Gyro Example VI. This VI displays the current angle of the gyroscope you specify.
9. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to display the block diagram.

Note You might need to resize the While Loop and clear some space below the Quotient & Remainder function on the block diagram of the Multiple Data Gyro Example VI before completing the following steps.

10. Place a Bundle By Name function below the **Angle** indicator on the block diagram of the Gyro Example VI.
11. Resize the Bundle By Name function to display two elements.
12. Select **Select a VI** on the **Functions** palette. Navigate to the **myTypeDef** control you created and click the **OK** button.
13. Place the **myTypeDef** control below the Quotient & Remainder function.

Note You also can place the **myTypeDef** control on the block diagram by dragging the control icon from the upper right corner of the **Control Editor** window to the block diagram of the Multiple Data Gyro Example VI.

14. Wire the **myTypeDef** control to the **input cluster** input of the Bundle By Name function.
15. Click the second **Numeric** element of the Bundle By Name function and select **String** from the shortcut menu.

The affected portion of the block diagram should appear similar to the following figure.

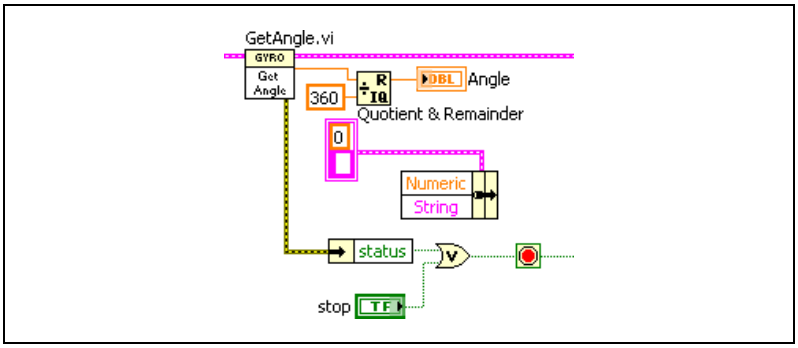


Figure 6-5. Bundling Elements for the myTypeDef Strict Type Definition

16. Wire the **x-y*floor(x/y)** output of the Quotient & Remainder function to the **Numeric** element of the Bundle By Name function.
17. Place a string constant to the left of the Bundle By Name function and enter `Example`.
18. Wire the string constant to the **String** input of the Bundle By Name function. You now can send both the gyroscope angle and this string to the Dashboard Main VI.
19. Place a Flatten To String function to the right of the Bundle By Name function.
20. Wire the **output cluster** output of the Bundle By Name function to the **anything** input of the Flatten To String function.
21. Place a String To Byte Array function to the right of the Flatten To String function.
22. Wire the **data string** output of the Flatten To String function to the **string** input of the String To Byte Array function.

23. Place a Set User Data VI to the right of the String To Byte Array function. This VI specifies the user data that the Dashboard Main VI receives.
24. Wire the **unsigned byte array** output of the String To Byte Array function to the **User Data (984 bytes)** input of the Set User Data VI.

The affected portion of the block diagram should appear similar to the following figure.

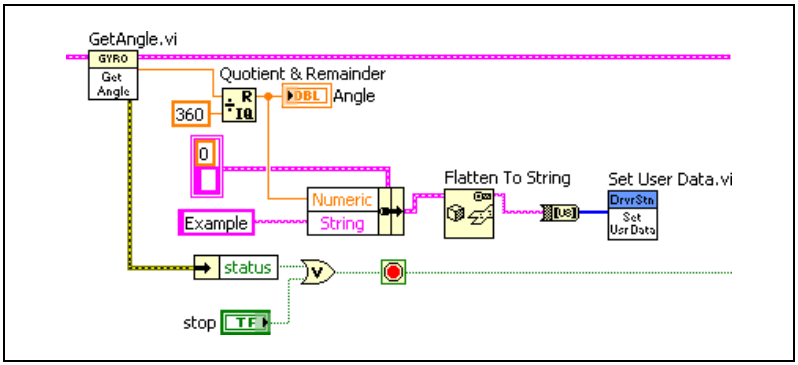


Figure 6-6. Sending Multiple Data Values to the Dashboard Main VI

25. Select **File»Save As** and save the VI as `Multiple Data Gyro Example.vi` in an easily accessible location.

The default Gyro Example VI returns the gyroscope angle as a double-precision, floating-point data type and does not return any string data. The modified VI converts the gyroscope angle and string data to an array of bytes, which the Set User Data VI can accept. The Set User Data VI then sends these multiple data values to the Dashboard Main VI.

Adding Indicators to the Dashboard Main VI

The Multiple Data Gyro Example VI now sends the gyroscope data to the Dashboard Main VI. This section describes how to set up the Dashboard Main VI to receive and display the user data. Complete the following steps to display multiple data values from the Multiple Data Gyro Example VI in the Dashboard Main VI.

1. Select **View»Getting Started Window** to display the **Getting Started** window.
2. Click the **FRC Dashboard Project** link in the **Getting Started** window to display the **Create New FRC Dashboard Project** dialog box.

3. Enter `FRC Dashboard Project Multiple Data Example` in the **Project name** text box of the **Create New FRC Dashboard Project** dialog box.
4. Click the **Finish** button in the **Create New FRC Dashboard Project** dialog box to create a new FRC dashboard project.
5. Double-click the **Dashboard Main.vi** item in the **Project Explorer** window to open the Dashboard Main VI.
6. Place a **Gauge** indicator on the front panel below the **User Data** section of the front panel.
7. Right-click the **Gauge** indicator and select **Properties** from the shortcut menu to display the **Knob Properties** dialog box.
8. On the **Appearance** page, enter `Gyroscope Angle` in the **Label Text** text box to change the name of the indicator.
9. On the **Scale** page, enter `360` as the **Maximum** value in the **Scale Range** section. The **Gyroscope Angle** indicator now displays the heading of the gyroscope within a range of 360 degrees.
10. Click the **OK** button to apply the new value.
11. Hover over the 360 marker on the **Gyroscope Angle** indicator and use the Operating tool to drag the marker to the 0 value. When you drag a marker, the cursor changes to a circular arrow to indicate the tool is over a marker.
12. Right-click the **Gyroscope Angle** indicator and select **Visible Items»Digital Display** from the shortcut menu to add a digital display to the **Gauge** indicator.
13. Place a **String** indicator above the **Gyroscope Angle** indicator.
14. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to display the block diagram.

Note You might need to expand the second While Loop and clear some space to the right of the **Battery level** indicator on the block diagram of the Dashboard Main VI before completing the following steps.

15. Place the **Gyroscope Angle** and **String** indicators in the second While Loop of the block diagram to the right of the **Battery level** indicator.
16. Place an Unbundle function between the **Battery level** indicator and the **Gyroscope Angle** and **String** indicators.
17. Wire the **Strings** output of the Receive DS Packet VI to the **cluster** input of the Unbundle function.

18. Select **Select a VI** on the **Functions** palette. Navigate to the **myTypeDef** control you created and click the **OK** button.
19. Place the **myTypeDef** above the Unbundle function.
20. Place an Unflatten From String function between the Unbundle function and the **Gyroscope Angle** and **String** indicators.
21. Wire the **myTypeDef** constant to the **type** input of the Unflatten From String function.
22. Wire the **User Data** output of the Unbundle function to the **binary string** input of the Unflatten From String function.
23. Place an Unbundle By Name function between the Unflatten From String function and the **Gyroscope Angle** and **String** indicators. Resize the Unbundle By Name function to display two elements.
24. Wire the **value** output of the Unflatten From String function to the **input cluster** input of the Unbundle By Name function.
25. Click the second **Numeric** element of the Unbundle By Name function and select **String** from the shortcut menu.
26. Wire the **Numeric** element of the Unbundle By Name function to the **Gyroscope Angle** indicator.
27. Wire the **String** element of the Unbundle By Name function to the **String** indicator.

The affected portion of the block diagram should appear similar to the following figure.

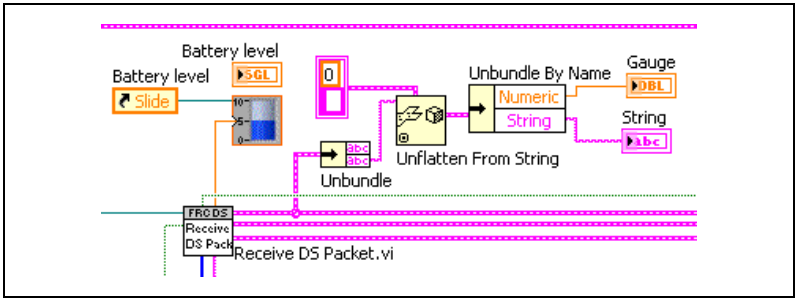


Figure 6-7. Displaying Multiple Data Values in the Dashboard Main VI

28. Select **File»Save As** and save the VI as `Multiple Data Dashboard Main.vi` in an easily accessible location.

The Multiple Data Dashboard Main VI now receives the gyroscope angle and string data from the Multiple Data Gyro Example VI and displays the data in the **User Data** section of the front panel.

Running the Dashboard Main VI

Complete the following steps to run the Multiple Data Gyro Example and Multiple Data Dashboard Main VIs.

1. Switch to the Multiple Data Gyro Example VI.
2. Verify that you configured the IP address of the CompactRIO device correctly. Refer to the *Configuring the CompactRIO Device* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the IP address of the CompactRIO device.
3. Click the **Run** button to deploy the Multiple Data Gyro Example VI to the CompactRIO device. The Multiple Data Gyro Example VI runs on the CompactRIO device and sends both the angle and string data to the Dashboard Main VI.
4. Switch to the Multiple Data Dashboard Main VI.
5. Verify that you configured the IP address of the host computer correctly. Refer to the *Setting the Static IP Address of the Computer* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the IP address of the host computer.
6. Click the **Run** button to run the Multiple Data Dashboard Main VI on the host computer. This VI now reflects the changes in the angle of the gyroscope and shows the **Example** string value.
7. Stop and close all VIs and projects.

This tutorial demonstrated how to send multiple data values to the Dashboard Main VI. Use the techniques in this tutorial to display any user data in the Dashboard Main VI.

Refer to the *FRC Dashboard Project* section of Chapter 4, *Using the FRC Framework*, for more information about the FRC dashboard project.

7

Troubleshooting the FRC Robot

In previous chapters, you learned how to program a robot using the CompactRIO device, the *FIRST* Robotics Competition framework, and the WPI Robotics Library VIs. However, at times, the robot might not work as you expect. Table 7-1 describes common issues you might encounter when working with the robot and solutions to address those issues.

Table 7-1. Common Programming Issues

Issue	Solution
The CompactRIO Imaging Tool does not run on some computers with two Internet connections.	Disable one of the Internet connections and run the CompactRIO Imaging Tool again. For example, if a computer has both a wireless Internet connection and a wired connection, disable the wireless connection and run the CompactRIO Imaging Tool again. Refer to the *Running the CompactRIO Imaging Tool* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the CompactRIO device.
The motors on the FRC robot do not run when I attempt to run them.	Ensure that the user watchdog is enabled and that you configured the Watchdog VIs to feed the user watchdog. You can use the SetEnabled VI to specify whether the user watchdog is enabled. If you enable the user watchdog, you must specify a timeout period and ensure that the program you write feeds the watchdog regularly. The user watchdog is enabled by default. If you do not feed the user watchdog during the period you specify, the watchdog disables the PWM, relay, and solenoid outputs of the CompactRIO device. Refer to Chapter 42, *Watchdog VIs*, for more information about the WPI Robotics Library Watchdog VIs. Additionally, use the Start Communication VI to set up communications between the driver station and the CompactRIO. Ensure that the program that you run on the CompactRIO includes this VI. Otherwise, the system watchdog shuts down the motors. Refer to Chapter 23, *DriverStation VIs*, for more information about the DriverStation VIs.

Table 7-1. Common Programming Issues (Continued)

Issue	Solution
When I try to deploy a program to the CompactRIO device, the program does not run. Sometimes the FRC robot begins operation although I did not run a program.	Launch the **CompactRIO Imaging Tool** dialog box and examine the **Choose Development Environment** section. Ensure that you select the development environment in which you want to work, then redeploy the program. If you specify a development environment different from the one in which you want to work, the program from the other development environment might run instead and prevent you from accessing the CompactRIO device. Refer to the *Running the CompactRIO Imaging Tool* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about the CompactRIO Imaging Tool.
I cannot access the Axis camera using the *FIRST* Vision VIs.	Ensure the Axis camera is connected to the CompactRIO device using the orange Ethernet crossover cable in the FRC kit. Refer to Chapter 3, *Configuring the Camera and the CompactRIO Device*, for more information about configuring the Camera. Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for more information about the *FIRST* Vision VIs.
When I try to run the CompactRIO Imaging Tool, the **CompactRIO Imaging Tool** dialog box does not list any CompactRIO devices connected to the host computer.	Check the network firewall and ensure that the firewall allows the computer to access the CompactRIO device. You might need to disable the firewall for the CompactRIO Imaging Tool to run with no errors.
When I run the CompactRIO Imaging Tool, the tool stops while downloading an image to the CompactRIO device.	Ensure that the network firewall allows the computer to access the CompactRIO device with both an `0.0.0.0` IP address and an `10.`*`xx.yy`*`.02` IP address. During the imaging process, the CompactRIO Imaging Tool sets the IP address of the CompactRIO device to `0.0.0.0`. You might need to disable the firewall for the CompactRIO Imaging Tool to run with no errors.

8

Using the WPI Robotics Library VIs

Use the WPI Robotics Library VIs to interface with the CompactRIO device and perform tasks such as reading and writing data to sensors and driving motors.

Refer to Chapters 9 through 42 for reference information about the WPI Robotics Library VIs. Each palette of VIs is listed in alphabetical order.

Reference Clusters

Many of the WPI Robotics Library VIs contain input and output reference clusters, such as **CompressorDevRef**, **RelayDevRef**, and so on. Use these reference clusters to pass information about a specific sensor or module between VIs.

For example, the following figure illustrates how to open a reference to an encoder, start the same encoder, stop the encoder, and then close the corresponding reference.

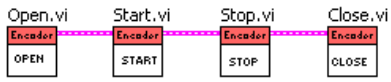


Figure 8-1. Using Reference Clusters with the Encoder VIs

First, use the Open VI on the **Encoder** palette to open a reference to an encoder. Opening a reference for an encoder reserves an available encoder index for that encoder. If all encoder indexes are reserved already, you cannot open a reference for the encoder until you close an existing encoder reference. The Open VI returns an **EncoderDevRef** output reference cluster that identifies the encoder for which you opened the reference. Wire the **EncoderDevRef** output reference cluster to the **EncoderDevRef** input reference cluster of the Start VI to specify this encoder as the one you want to start. Similarly, wire the **EncoderDevRef** output reference cluster of the Start VI to the **EncoderDevRef** input reference of the Stop VI to specify the same encoder as the one you want to stop. Finally, wire the

EncoderDevRef output reference cluster of the Stop VI to the **EncoderDevRef** input reference cluster of the Close VI to close the corresponding reference.

Wiring the **EncoderDevRef** reference cluster between VIs establishes a reference to the same encoder for each VI. By using the reference cluster, you do not have to specify the same information about the encoder for each VI.

Caution Do *not* manually specify any information in a reference cluster. Always use a corresponding Open VI for the sensor or module to create the reference cluster that you then can wire to other VIs.

All input and output reference clusters contain at least a **DevStatus** cluster, which contains error information and is similar to the LabVIEW error cluster. Refer to the *Getting Started with LabVIEW for the FIRST Robotics Competition* manual, available by navigating to the `National Instruments\LabVIEW 8.5\manuals` directory and opening `FRC_Getting_Started.pdf`, for more information about the LabVIEW error cluster.

Some reference clusters also contain additional controls or indicators unique to the sensor or module to which they apply. For example, the **EncoderDevRef** reference cluster, shown as follows, contains a **DevStatus** cluster as well four encoder-specific controls: **EncoderIndex**, **CounterIndex**, **DistancePerCount**, and **Decoding Type**.

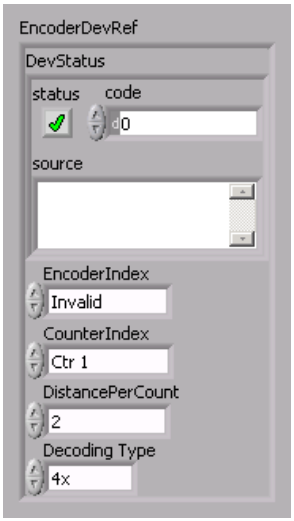


Figure 8-2. EncoderDevRef Input Reference Cluster

EncoderIndex specifies the index of the reserved encoder. Therefore, **EncoderIndex** establishes a reference to a particular encoder.

Error Handling

When you use reference clusters to connect VIs, you also pass error information between those VIs. You can use this error information to troubleshoot the application.

In Figure 8-1, *Using Reference Clusters with the Encoder VIs*, if the Open VI runs normally, the **DevStatus** cluster of the **EncoderDevRef** output reference cluster is empty. The Open VI therefore does not pass any errors to the Start VI, and the **DevStatus** cluster of the **EncoderDevRef** input reference cluster of the Start VI also is empty.

However, suppose an error occurs when the Start VI runs. The Start VI returns an error in the **DevStatus** cluster of the **EncoderDevRef** output reference cluster and passes this information to the **EncoderDevRef** input reference cluster of the Stop VI. Because the Stop VI receives an error, it does not execute and passes the error information to the Close VI, again through the **EncoderDevRef** reference cluster. If you wire an indicator to the **error out** output of the Close VI, **error out** returns the cumulative error information for all VIs preceding and including the Close VI. From this error information, you can determine that an error originated with the Start VI, and you can troubleshoot that error accordingly.

Many of the WPI Robotics Library VIs also contain **error in** and **error out** clusters. You can use these clusters to merge error information from different parts of an application. Each WPI Robotics Library VI merges the error information it receives from the input reference cluster and the **error in** cluster and returns this merged information in both the output reference cluster and the **error out** cluster.

9

Accelerometer VIs

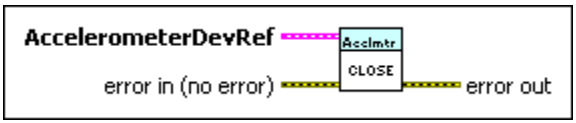
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Accelerometer VIs.

Use the Accelerometer VIs to determine the acceleration of a robot as measured by an accelerometer. Accelerometers measure both dynamic acceleration, or vibration, and static acceleration, or gravity. The accelerometer that the *FIRST* Robotics Competition (FRC) kit provides is a two-axis accelerometer. This accelerometer provides acceleration data in the x-axis and y-axis relative to the circuit board. You also can use the accelerometer as a tilt sensor to measure the acceleration of gravity.

Use the Open VI to create an **AccelerometerDevRef** reference cluster that you then can wire to the other Accelerometer VIs.

Close.vi

Closes the reference to the accelerometer you specify. Use this VI to close each accelerometer reference that you open with the Open VI.



AccelerometerDevRef specifies a reference to the accelerometer you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Scaling sets the sensitivity of the accelerometer that you specify.

Gain (VoltsPerG) sets the amount of volts per gravity of acceleration. The default value is 1.

Center Voltage sets the offset portion of the scaling to gravities. The default value is 2.5.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select

Explain Error from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetAcceleration.vi

Returns the current acceleration of the sensor that you specify.

AccelerometerDevRef specifies a reference to the accelerometer you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Scaling sets the sensitivity of the accelerometer that you specify.

Gain (VoltsPerG) sets the amount of volts per gravity of acceleration. The default value is 1.

Center Voltage sets the offset portion of the scaling to gravities. The default value is 2.5.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AccelerometerDevRef returns a reference to the accelerometer.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Scaling returns the sensitivity of the accelerometer.

Gain (VoltsPerG) returns the amount of volts per gravity of acceleration.

Center Voltage returns the offset portion of the scaling to gravities.

Acceleration returns the acceleration in floating point units of gravities.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

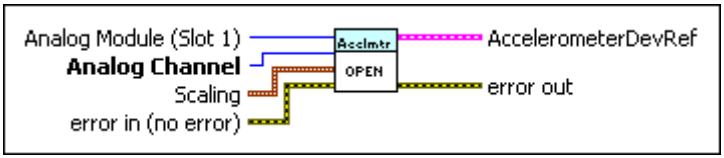
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the accelerometer you specify. You must open a reference before using any other VIs on this palette.



Analog Module (Slot 1) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **Slot 1** or **Slot 2**. The default is **Slot 1**.

Analog Channel specifies the channel of the **Analog Module** you want to use. Select a value between **AI 1** and **AI 8**. The default is **AI 1**. If **Analog Channel** is **Invalid**, this VI returns an error.

Scaling sets the sensitivity of the accelerometer that you specify.

Gain (VoltsPerG) sets the amount of volts per gravity of acceleration. The default value is 1.

Center Voltage sets the offset portion of the scaling to gravities. The default value is 2.5.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AccelerometerDevRef returns a reference to the accelerometer.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Scaling returns the sensitivity of the accelerometer.

Gain (VoltsPerG) returns the amount of volts per gravity of acceleration.

Center Voltage returns the offset portion of the scaling to gravities.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

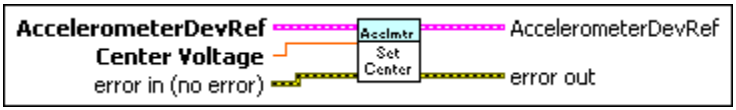
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetCenterVoltage.vi

Sets the offset portion of the scaling to gravities of acceleration.



AccelerometerDevRef specifies a reference to the accelerometer you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Scaling sets the sensitivity of the accelerometer that you specify.

Gain (VoltsPerG) sets the amount of volts per gravity of acceleration. The default value is 1.

Center Voltage sets the offset portion of the scaling to gravities. The default value is 2.5.

Center Voltage sets the offset portion of the scaling to gravities. This parameter overwrites the value in the **Center Voltage** subparameter of the **AccelerometerDevRef** input.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AccelerometerDevRef returns a reference to the accelerometer.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Scaling returns the sensitivity of the accelerometer.

Gain (VoltsPerG) returns the amount of volts per gravity of acceleration.

Center Voltage returns the offset portion of the scaling to gravities.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

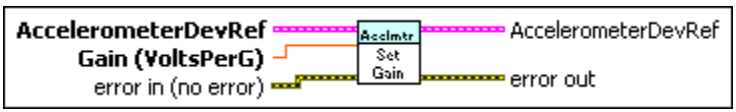
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetGain.vi

Specifies the voltage used per degree of rotation per second. This VI specifies one of the parameters used by the GetAngle VI to calculate and report the heading of the robot.



AccelerometerDevRef specifies a reference to the accelerometer you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Scaling sets the sensitivity of the accelerometer that you specify.

Gain (VoltsPerG) sets the amount of volts per gravity of acceleration. The default value is 1.

Center Voltage sets the offset portion of the scaling to gravities. The default value is 2.5.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AccelerometerDevRef returns a reference to the accelerometer.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Scaling returns the sensitivity of the accelerometer.

Gain (VoltsPerG) returns the amount of volts per gravity of acceleration.

Center Voltage returns the offset portion of the scaling to gravities.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

10

Accumulator VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Accumulator VIs.

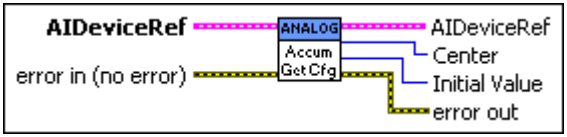
Use the Accumulator VIs to configure and access accumulators on the CompactRIO device. An accumulator is a register on the FPGA that you can use to store cumulative results. You can access values from the accumulator faster than you can access values in main memory.
One common use of accumulators is to perform averaging on signals.

The FPGA of the CompactRIO device contains two accumulators. You can access these accumulators only through the analog module in slot 1 of the CompactRIO device.

Use the AnalogChannel Open VI to create an **AIDeviceRef** reference cluster that you then can wire to the Accumulator VIs.

GetConfiguration.vi

Returns the center value and initial value of the accumulator on the FPGA on the CompactRIO device. The FPGA subtracts the center value from the value of each element before adding the value of the element to the cumulative value of the accumulator. Use the center value and initial value to account for offset in devices, such as gyros and accelerometers, when integrating signals from these devices.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Initial Value returns the initial value of the accumulator. The default value is 0.

Center returns the center value of the accumulator. The value of the center value depends on the output of channel 1 on the module. Therefore the center value is different depending on whether you sample or oversample the signal on channel 1.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

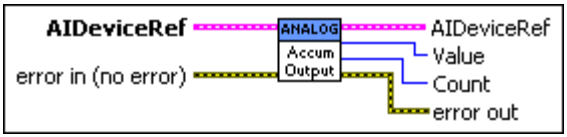
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetOutput.vi

Returns the current value of the accumulator on the FPGA on the CompactRIO device. The GetOutput VI also returns the number of elements accumulated. You can use this VI to perform averaging.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between **`AI 1`** and **`AI 8`**. If **Analog Channel** returns a value of **`Invalid`**, the VI did not find the analog channel you specified, and this VI returns an error.

Value returns the current value of the accumulator.

Count returns the number of elements in the accumulator.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

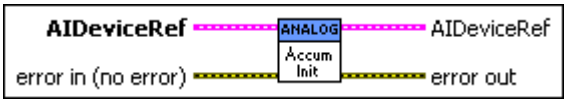
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Init.vi

Initializes the accumulator on the FPGA on the CompactRIO device. Initializing the accumulator sets the center value of the accumulator to zero and sets the cumulative accumulator value to zero.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

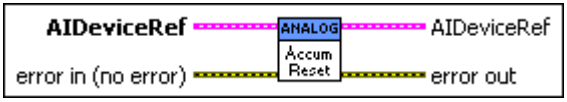
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Reset.vi

Resets the accumulator on the FPGA on the CompactRIO device to the initial value you set with the SetConfiguration VI. By default, the initial value is 0.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

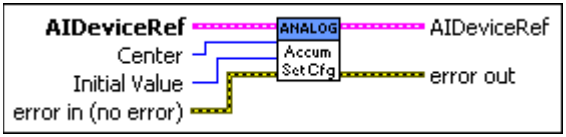
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetConfiguration.vi

Sets the center value and initial value of the accumulator on the FPGA on the CompactRIO device. The FPGA subtracts the center value from the value of each element before adding the value of the element to the cumulative value of the accumulator. Set the center value and initial value to account for offset in devices, such as gyros and accelerometers, when integrating signals from these devices.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Center specifies the center value of the accumulator. The value of the center depends on the output of channel 1 on the module. Therefore the center value is different depending on whether you sample or oversample the signal on channel 1.

Initial Value specifies the initial value of the accumulator. The default value is 0.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between **`AI 1`** and **`AI 8`**. If **Analog Channel** returns a value of **`Invalid`**, the VI did not find the analog channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

11

Actuators VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Actuator VIs.

Use the Actuators VIs to control the behavior of actuators such as limit switches, motors, and solenoids.

12

AnalogChannel VIs

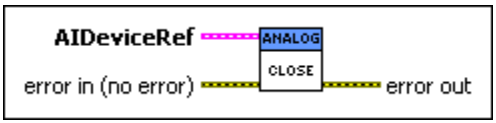
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the AnalogChannel VIs.

Use the AnalogChannel VIs to acquire an analog signal from a channel on a module on the CompactRIO device. You can configure how the channel samples the signal by specifying the number of bits and the number of oversample bits. You also can acquire the scaled voltage signal and the average scaled voltage signal.

Use the Open VI to create an **AIDeviceRef** reference cluster that you then can wire to the other AnalogChannel VIs.

Close.vi

Closes the reference to the analog channel you specify. Use this VI to close each analog channel reference that you open with the Open VI.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

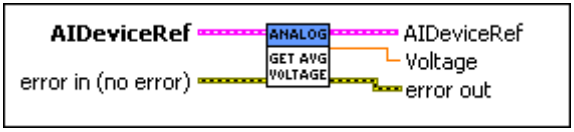
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetAverageVoltage.vi

Returns the average voltage value from the oversample and average engine for the channel. The converter scales the raw value to the voltage value using the least significant bit weight constant and the offset constant of the hardware. Oversampling increases the resolution of the voltage value but decreases the update rate. Averaging returns a stable voltage value but also decreases the update rate. Use the GetVoltage VI to get voltage values directly from the A/D converter.

You can use the GetLSBWeight VI and GetOffset VI to determine the least significant bit weight and offset constant for the module.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Voltage returns the voltage from the A/D converter.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

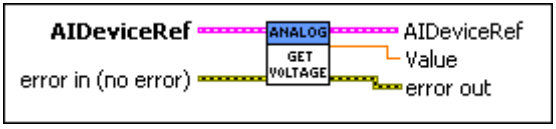
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetVoltage.vi

Returns the voltage value directly from the A/D converter on the channel. The converter scales the raw value to the voltage value using the least significant bit weight constant and the offset constant of the hardware. Use the GetAverageVoltage VI to get voltage values from oversampled or averaged signals.

You can use the GetLSBWeight VI and GetOffset VI to determine the least significant bit weight and offset constant for the module.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Value returns the voltage value of the signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

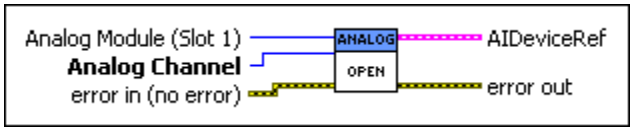
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the analog channel you specify. You must open a reference before using any other AnalogChannel VIs.



Analog Module (Slot 1) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **Slot 1** or **Slot 2**. The default is **Slot 1**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

13

AnalogChannel Advanced VIs

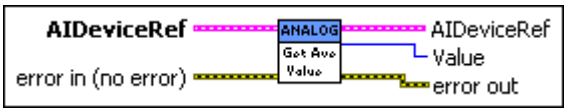
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the AnalogChannel Advanced VIs.

Use the AnalogChannel Advanced VIs to perform more advanced operations on analog signals. You can configure the sampling rate and set the averaging and oversampling bits. You also can get factory calibration data from the EEPROM on the module on the CompactRIO device.

Use the AnalogChannel Open VI to create an **AIDeviceRef** reference cluster that you then can wire to the AnalogChannel Advanced VIs.

GetAverageValue.vi

Returns the average value of the unscaled analog-to-digital converter (ADC) codes directly from the FPGA. The ADC codes are the binary values that the analog to digital converter returns. The GetAverageVoltage VI references this VI to scale the obtained value into volts.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Value returns the average value of the unscaled ADC codes directly from the FPGA.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

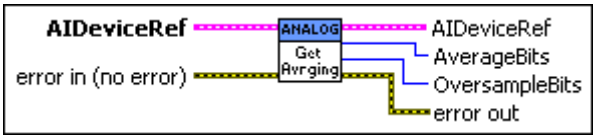
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetAveraging.vi

Returns the number of bits the A/D converter uses to average the signal and the number of bits the converter uses to oversample the signal. Use the SetAveraging VI to set the number of bits to average and oversample.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

AverageBits returns the computed average of a number of bits from the A/D converter.

OversampleBits returns the number of bits the A/D converter uses for oversampling the signal. The number of oversample values is 2^*n*, where *n* is **OversampleBits**.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

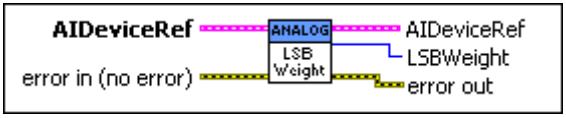
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetLSBWeight.vi

Returns the least significant bit (LSB) weight constant that the A/D converter uses to scale the signal.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between **`AI 1`** and **`AI 8`**. If **Analog Channel** returns a value of **`Invalid`**, the VI did not find the analog channel you specified, and this VI returns an error.

LSBWeight returns the least significant bit weight constant. The least significant bit weight constant is part of the calibration data for the specific module on the CompactRIO device. The factory stores the least significant bit weight constant in the EEPROM of the module.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

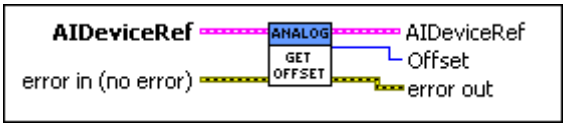
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetOffset.vi

Returns the factory offset value that the A/D converter uses to scale the signal.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Offset returns the amount that the A/D converter offsets the raw value of the signal to determine the voltage value. The offset constant is part of the calibration data for the specific module on the CompactRIO device. The factory stores the offset constant in the EEPROM of the module.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

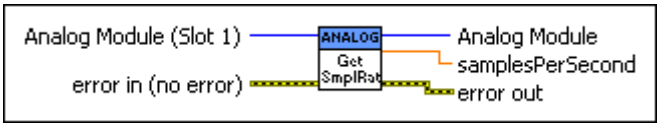
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetSampleRate.vi

Returns the sample rate of the analog module you specify. The effective sample rate for a channel on the module is the sample rate of the module divided by the number of active channels on the module. Use the SetSampleRate VI to set the sample rate of a channel on the module.



Analog Module (Slot 1) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **Slot 1** or **Slot 2**. The default is **Slot 1**.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

samplesPerSecond returns the sample rate of the module. The sample rate of a channel in the module is the module sample rate divided by the number of active channels.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetValue.vi

Returns the value of the unscaled analog-to-digital converter (ADC) codes directly from the FPGA. The ADC codes are the binary values that the analog to digital converter returns. The GetVoltage VI references this VI to scale the obtained value into volts.

AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Value returns the value of the unscaled ADC codes directly from the FPGA.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

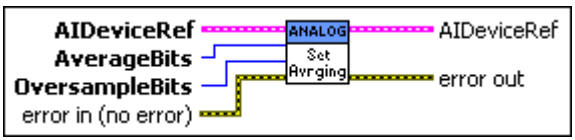
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetAveraging.vi

Specifies the number of bits you want to use to average and oversample the signal. Averaging can improve measurement accuracy for noisy and rapidly changing signals. Oversampling is a technique that sums extra samples but does not divide the sum by the number of samples. For example, if you oversample a signal by 16 times, the resulting values are 16 times larger than the average output. Use oversampling to improve the resolution of signal measurements at the expense of the sampling rate. The FPGA on the CompactRIO device automatically does the oversampling.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

AverageBits specifies the number of bits to use when computing the average of a signal.

OversampleBits specifies the number of bits to use for oversampling the signal. The number of oversample values is 2^*n*, where *n* is **OversampleBits**.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AIDeviceRef returns a reference to the analog channel.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

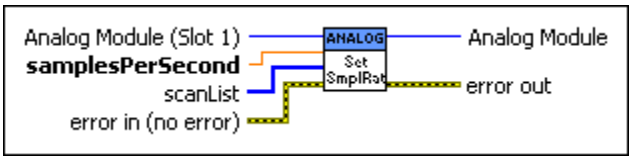
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetSampleRate.vi

Sets the sample rate for a channel on the module you specify. The sample rate is the same for each channel on the module.

You can use sampling theory to determine an optimal sampling rate for a signal.



Analog Module (Slot 1) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **Slot 1** or **Slot 2**. The default is **Slot 1**.

samplesPerSecond specifies the sample rate for the module.

scanList specifies the channel or channels of the analog module you want to make active for sampling. If the array is empty, the module samples all channels. Otherwise, the array must include elements for each channel to make active for sampling. You can make channels 1 through 8 active. If **scanList** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

14

AnalogTrigger VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the AnalogTrigger VIs.

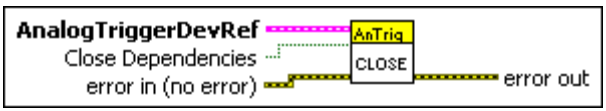
Use the AnalogTrigger VIs to create digital signals from analog signals. You can use analog triggers as sources for the Counter VIs and Encoder VIs. The analog trigger circuit on the target produces three types of output: TriggerState, InWindow, and Pulse. Use the AnalogTrigger Get Output VI to get the TriggerState and InWindow output from the trigger.

The FPGA on the CompactRIO device contains eight analog triggers. The analog triggers are not tied to a specific module or channel on the CompactRIO device. You can reserve and release analog triggers as needed.

Use the Open VI to create an **AnalogTriggerDevRef** reference cluster that you then can wire to the other AnalogTrigger VIs.

Close.vi

Closes the reference to the analog trigger you specify. Use this VI to close each analog trigger reference that you open with the Open VI.



AnalogTriggerDevRef specifies a reference to the analog trigger you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

Close Dependencies, when TRUE, closes any analog channel inputs associated with the analog trigger when you close the reference to the analog trigger.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

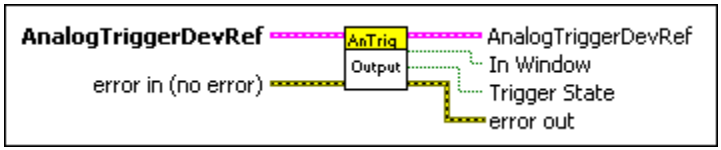
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetOutput.vi

Returns the different types of digital output from the trigger.



AnalogTriggerDevRef specifies a reference to the analog trigger you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerDevRef returns a reference to the analog trigger.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

AnalogTriggerIndex returns the index of the reserved analog trigger. **AnalogTriggerIndex** can return a value between **`Trig 0`** and **`Trig 7`**. If **AnalogTriggerIndex** returns a value of **`Invalid`**, the VI did not find the analog trigger you specified, and this VI returns an error.

In Window indicates, when TRUE, that the signal is between the upper and lower limits of the range you set in the VI.

Trigger State indicates, when TRUE, that the signal exceeds the upper limit you set in the VI. When **Trigger State** is FALSE, the signal is below the lower limit you set in the VI. When the signal is between the upper and lower limits, **Trigger State** maintains the most recent value.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

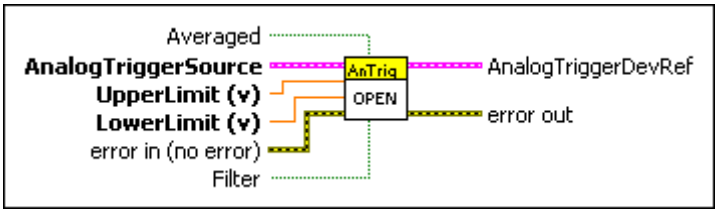
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens and configures a reference to the analog trigger. You must open a reference before using any other AnalogTrigger VIs.



Averaged specifies, when TRUE, that the analog source of the trigger contains averaged values.

AnalogTriggerSource specifies the source of the analog trigger you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

UpperLimit (v) specifies the upper limit of the voltage signal that determines the generation of the trigger.

LowerLimit (v) specifies the lower limit of the voltage signal that determines the generation of the trigger.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Filter, when TRUE, enables the Average Rejection Filter for the analog trigger you are opening. The Average Rejection Filter eliminates less reliable rising and falling pulses from the analog trigger you specify.

AnalogTriggerDevRef returns a reference to the analog trigger.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

AnalogTriggerIndex returns the index of the reserved analog trigger. **AnalogTriggerIndex** can return a value between `Trig 0` and `Trig 7`. If **AnalogTriggerIndex** returns a value of `Invalid`, the VI did not find the analog trigger you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

15

Camera VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Camera VIs.

Use the Camera VIs to acquire images with the Axis camera. You then can use the FIRST Vision VIs in the FRC robot project on the CompactRIO device to analyze and manipulate the images.

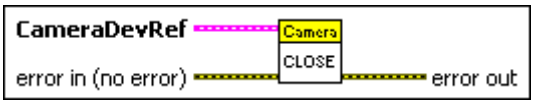
Use the Open VI to create a **CameraDevRef** reference cluster that you then can wire to the other Camera VIs.

You use most of the Camera VIs in the FRC robot project that you run on the CompactRIO device. For example, you open a reference to the camera with the Open VI, start acquiring image data with the Start VI, retrieve a specific image with the Get Image VI, stop acquiring image data with the Stop VI, and then close the reference to the camera with the Close VI. However, the Get Image From Controller VI must run on the host computer. The CompactRIO device continuously sends image data to the host computer as long as a reference to the camera on the CompactRIO device is open. Use the Get Image from Controller VI to retrieve a specific image on the host computer from the image data that the CompactRIO device sends.

Close.vi

Closes the reference to the camera, stops acquiring image data, and stops sending image data to the host computer. If the count of the camera reference is greater than one, this VI only decrements the count of the camera reference by one.

Each time you run the Open VI, you increment the count of the camera reference by one. Ensure you have a Close VI to decrement the count of the camera reference for each Open VI you use.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Image.vi

Retrieves a specific image from the image data that the camera you specify acquired. Use the Start VI to start acquiring image data from the camera before you use the Get Image VI to retrieve an image. When you run this VI, this VI retrieves the most current snapshot of the image data from the camera.

This VI runs in the FRC robot project on the CompactRIO device and retrieves an image on the CompactRIO device. Use the Get Image From Controller VI in the FRC dashboard project on the host computer to retrieve an image on the host computer.

Refer to Chapter 4, *Using the FRC Framework*, for more information about the FRC robot and dashboard projects.

CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Image specifies a reference into which this VI copies the image it retrieves.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Image Out returns the reference into which this VI copied the image it retrieved.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Image From Controller.vi

Retrieves a specific image from the image data that the camera connected to the CompactRIO device you specify acquired. Use this VI in the FRC dashboard project on the host computer.

Use an Open VI in the FRC robot project to open a reference to the camera. The Open VI initiates an asynchronous task that waits for a request from the host computer to retrieve the image data. Then use the Get Image From Controller VI to make the request to retrieve the image data.

Refer to Chapter 4, *Using the FRC Framework*, for more information about the FRC dashboard and robot projects.

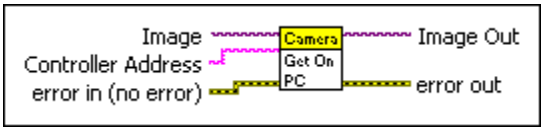


Image specifies a reference into which this VI copies the image it retrieves.

Controller Address specifies the IP address of the CompactRIO device from which you want to receive image data. The IP address of the CompactRIO device should be `10.xx.yy.2`, where *y* corresponds to the last two digits of your team number and *x* corresponds to the first or first two digits of your team number.

Refer to the *Configuring the CompactRIO Device* section of Chapter 3, *Configuring the Camera and the CompactRIO Device*, for information about using the CompactRIO Imaging Tool to configure the IP address of the CompactRIO device.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Image Out returns the reference into which this VI copied the image it retrieved.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

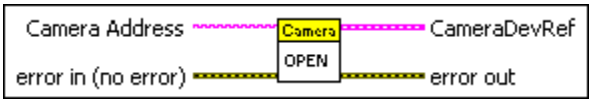
source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the camera you specify. If a reference to the camera is already open, this VI increments the count of the camera reference by one. You must open a reference before using any other VIs on this palette.

After you open a reference to the camera, you can use the Start and Stop VIs to start and stop acquiring image data, respectively, on the CompactRIO device. You also can use the Get Image From Controller VI in the FRC dashboard project to retrieve images on the host computer.

Each time you run the Close VI, you decrement the count of the camera reference by one. Ensure you have a Close VI to decrement the count of the camera reference for each Open VI you use.



Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error**

out. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef returns a reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Start.vi

Starts acquiring image data from the camera you specify. If an acquisition is already in progress, this VI increments the number of acquisition requests by one. After you start acquiring image data from the camera, you can use the Get Image VI in the FRC robot project to retrieve a specific image on the CompactRIO device.

Each time you run the Stop VI, you decrement the number of acquisition requests by one. Ensure you have a Stop VI to decrement the count of acquisition requests for each Start VI you use.

Refer to Chapter 4, *Using the FRC Framework*, for more information about the FRC robot and dashboard projects.

CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

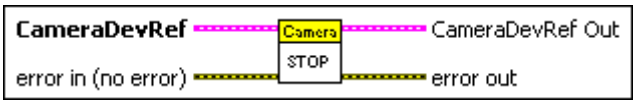
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Stop.vi

Stops acquiring image data from the camera you specify. If the count of acquisition requests is greater than one, this VI only decrements the number of acquisition requests by one.

Each time you run the Start VI, you increment the number of acquisition requests by one. Ensure you have a Stop VI to decrement the count of acquisition requests for each Start VI you use.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

16

Camera Properties VIs

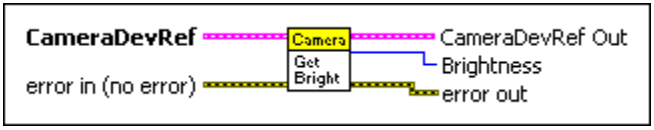
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Camera Properties VIs.

Use the Camera Properties VIs to set and return settings for the Axis camera.

Use the Camera Open VI to create a **CameraDevRef** reference cluster that you then can wire to the Camera Properties VIs.

Get Brightness.vi

Returns the brightness setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Brightness returns the brightness setting of the camera. **Brightness** can return a value between 0 and 100, where 0 indicates the darkest setting and 100 indicates the brightest setting.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

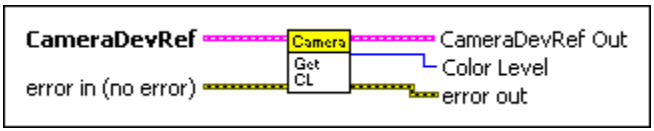
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Color Level.vi

Returns the color level setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Color Level returns the color level setting of the camera. **Color Level** can return a value between 0 and 100.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Exposure.vi

Returns the exposure setting of the camera you specify.

CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Exposure returns the exposure setting of the camera.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

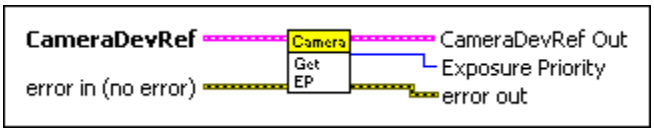
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Exposure Priority.vi

Returns whether the camera you specify prioritizes the frame rate or image quality when setting the exposure.

If the camera prioritizes the frame rate, the amount of noise in the image might increase. If the camera prioritizes image quality, the frame rate might decrease, resulting in increased motion blur.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Exposure Priority returns whether the camera prioritizes the frame rate or image quality when setting the exposure.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

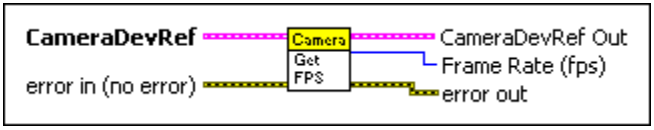
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Frame Rate.vi

Returns the frequency, in frames per second, at which the camera you specify captures a new set of image data.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Frame Rate (fps) returns the frequency, in frames per second, at which the camera captures a new set of image data.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

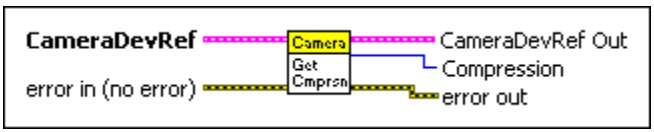
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Image Compression.vi

Returns the level of compression of the image from the camera. The higher the level of image compression, the smaller the file size but the lower the image quality.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Compression returns the level of image compression.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

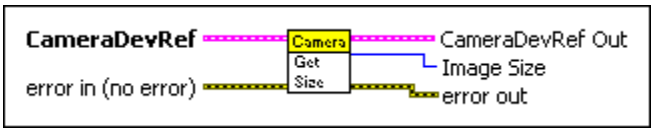
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Image Size.vi

Returns the size of the image the camera you specify acquired.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Image Size returns the size, in pixels, of the image.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

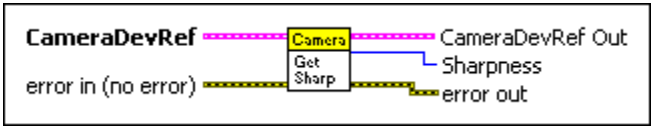
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get Sharpness.vi

Returns the sharpness setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Sharpness returns the sharpness setting of the camera. **Sharpness** can return a value between 0 and 100, where 0 indicates the least sharp setting and 100 indicates the sharpest setting.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

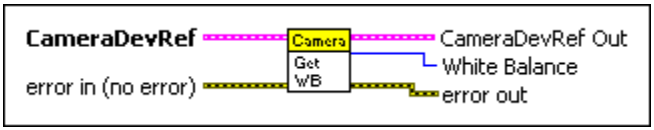
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get White Balance.vi

Returns the white balance setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

White Balance returns the white balance setting of the camera.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

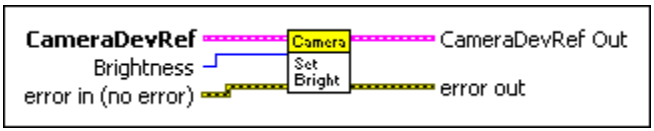
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Brightness.vi

Specifies the brightness setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Brightness specifies the brightness setting of the camera. You can specify a value between 0 and 100, where 0 specifies the darkest setting and 100 specifies the brightest setting. The default is 50.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

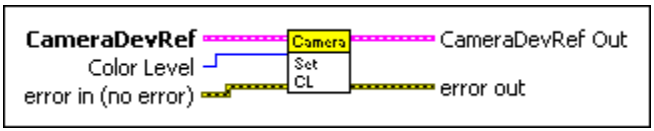
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Color Level.vi

Specifies the color level setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Color Level specifies the color level setting of the camera. You can specify a value between 0 and 100. The default is 50.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

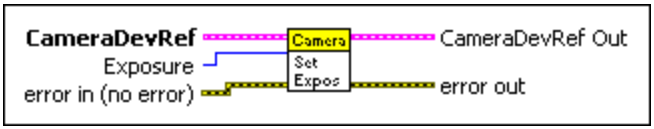
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Exposure.vi

Specifies the exposure setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Exposure specifies the exposure setting of the camera.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

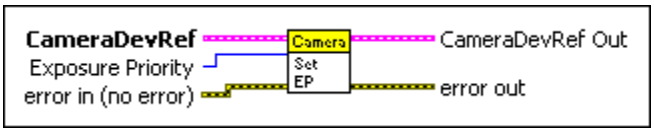
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Exposure Priority.vi

Specifies whether the camera you specify prioritizes the frame rate or image quality when setting the exposure.

If the camera prioritizes the frame rate, the amount of noise in the image might increase. If the camera prioritizes image quality, the frame rate might decrease, resulting in increased motion blur.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Exposure Priority specifies whether the camera prioritizes the frame rate or image quality when setting the exposure.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

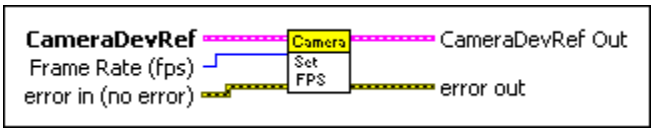
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Frame Rate.vi

Specifies the frequency, in frames per second, at which you want the camera you specify to capture a new set of image data.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Frame Rate (fps) specifies the frequency, in frames per second, at which the camera captures a new set of image data. The default is 30.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

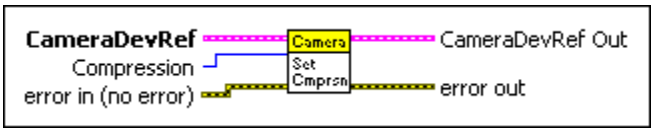
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Image Compression.vi

Sets the level of compression of the image from the camera. The higher the level of image compression, the smaller the file size but the lower the image quality.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Compression specifies the level of image compression. The default is 30.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

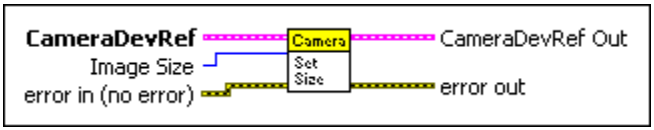
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Image Size.vi

Specifies the size of the image you want the camera you specify to acquire.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Image Size specifies the size, in pixels, of the image you want to acquire.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

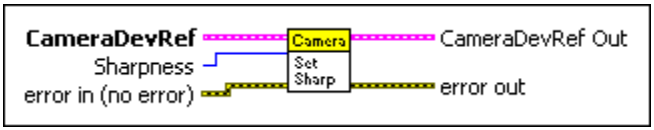
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Sharpness.vi

Specifies the sharpness setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Sharpness specifies the sharpness setting of the camera. You can specify a value between 0 and 100, where 0 specifies the least sharp setting and 100 specifies the sharpest setting. The default is 0.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

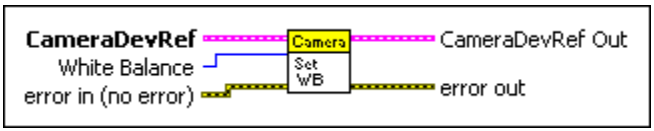
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set White Balance.vi

Specifies the white balance setting of the camera you specify.



CameraDevRef specifies a reference to the camera you want to use. Use the VI to open this reference.

Camera Address specifies the IP address of the camera you want to use. The default is `192.168.0.90`, which is the IP address to which the Axis camera is set by default.

CameraStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

White Balance specifies the white balance setting of the camera.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CameraDevRef Out returns the reference to the camera.

Camera Address returns the IP address of the camera.

CameraStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

17

Communications VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Communications VIs.

Use the Communications VIs to communicate with sensors, custom circuits, and other devices connected to the CompactRIO device.

18

Compressor VIs

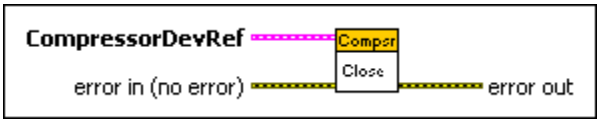
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Compressor VIs.

Use the Compressor VIs to start and stop the compressor of the robot. The Compressor VIs manipulate a pressure switch connected to the high pressure side of the pneumatic circuit of the robot.

Use the Open VI to create a **CompressorDevRef** reference cluster that you then can wire to the other Compressor VIs.

Close.vi

Closes the reference to the compressor you specify. Use this VI to close each compressor reference that you open with the Open VI. If you do not stop a running compressor with the Stop VI, closing the reference to the compressor stops the compressor.



CompressorDevRef specifies a reference to the compressor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

FIFO_enabled specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the compressor.

Enabled specifies, when TRUE, that the compressor is running. If **Enabled** is FALSE, the compressor is idle.

FIFO_relay specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the state of the compressor relay.

TaskVI specifies a reference to the RunCompressor subVI in the VI.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

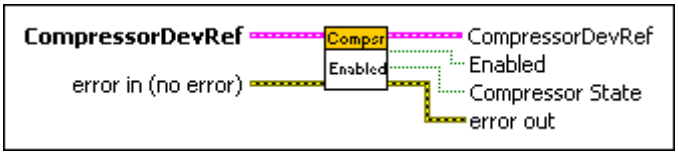
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetEnableState.vi

Returns the running state of the compressor and the on/off state of the compressor relay.



CompressorDevRef specifies a reference to the compressor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

FIFO_enabled specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the compressor.

Enabled specifies, when TRUE, that the compressor is running. If **Enabled** is FALSE, the compressor is idle.

FIFO_relay specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the state of the compressor relay.

TaskVI specifies a reference to the RunCompressor subVI in the VI.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CompressorDevRef returns a reference to the compressor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FIFO_enabled returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

Enabled returns TRUE if the compressor is running and FALSE if the compressor is idle.

FIFO_relay returns a reference to the FIFO. This parameter acts as a real-time status indicator of the state of the compressor relay.

TaskVI returns a reference to the RunCompressor subVI in the VI.

Enabled returns TRUE if the compressor is running and FALSE if the compressor is idle.

Compressor State returns TRUE if the compressor relay is on and FALSE if the compressor relay is off.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

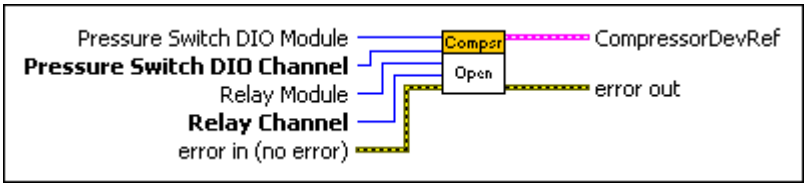
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the compressor you specify. You must open a reference before using any other VIs on this palette. After you open a reference to the compressor, you can use the Start and Stop VIs to start and stop the compressor, respectively.



Pressure Switch DIO Module specifies the digital module on the CompactRIO device to which the pressure switch is connected. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Pressure Switch DIO Channel specifies the channel on the **Pressure Switch DIO Module** to which the pressure switch is connected. Select a value between **DIO 1** and **DIO 14**. The default is **DIO 1**. If **Pressure Switch DIO Channel** is **Invalid**, this VI returns an error.

Relay Module specifies the slot number on the CompactRIO device of the digital module you want to use for the relay. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel specifies the channel of the **DIO Module** to which the relay is connected. Select a value between **Relay 1** and **Relay 8**. The default is **Relay 1**. If **Relay Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CompressorDevRef returns a reference to the compressor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FIFO_enabled returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

Enabled returns TRUE if the compressor is running and FALSE if the compressor is idle.

FIFO_relay returns a reference to the FIFO. This parameter acts as a real-time status indicator of the state of the compressor relay.

TaskVI returns a reference to the RunCompressor subVI in the VI.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

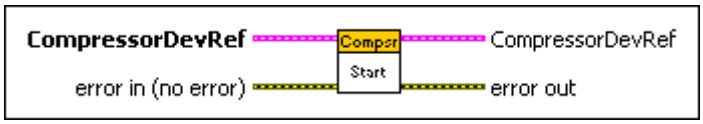
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Start.vi

Starts the compressor thread that you specify. Use the Open VI to open a reference to the compressor before you use this VI.



CompressorDevRef specifies a reference to the compressor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

FIFO_enabled specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the compressor.

Enabled specifies, when TRUE, that the compressor is running. If **Enabled** is FALSE, the compressor is idle.

FIFO_relay specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the state of the compressor relay.

TaskVI specifies a reference to the RunCompressor subVI in the VI.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CompressorDevRef returns a reference to the compressor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FIFO_enabled returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

Enabled returns TRUE if the compressor is running and FALSE if the compressor is idle.

FIFO_relay returns a reference to the FIFO. This parameter acts as a real-time status indicator of the state of the compressor relay.

TaskVI returns a reference to the RunCompressor subVI in the VI.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

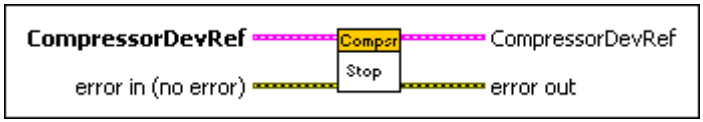
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Stop.vi

Stops the compressor that you specify.



CompressorDevRef specifies a reference to the compressor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

FIFO_enabled specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the compressor.

Enabled specifies, when TRUE, that the compressor is running. If **Enabled** is FALSE, the compressor is idle.

FIFO_relay specifies a reference to a real-time FIFO and overwrites the oldest data element when the FIFO is full. This parameter acts as a real-time status control of the state of the compressor relay.

TaskVI specifies a reference to the RunCompressor subVI in the VI.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CompressorDevRef returns a reference to the compressor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FIFO_enabled returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

Enabled returns TRUE if the compressor is running and FALSE if the compressor is idle.

FIFO_relay returns a reference to the FIFO. This parameter acts as a real-time status indicator of the state of the compressor relay.

TaskVI returns a reference to the RunCompressor subVI in the VI.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

19

Counter VIs

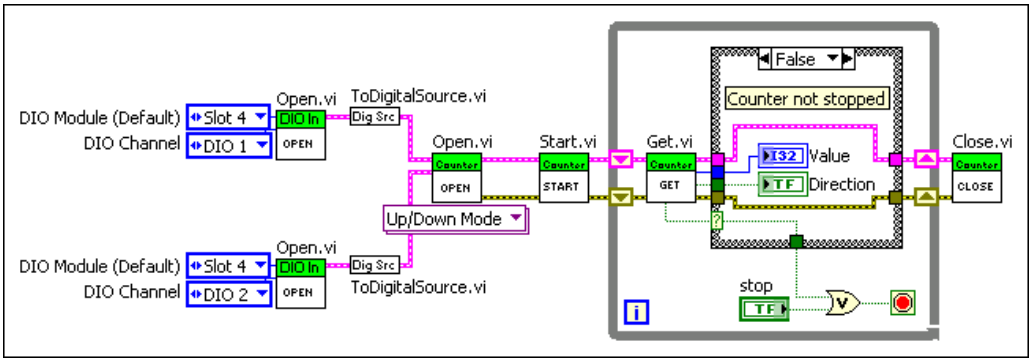
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Counter VIs.

Use the Counter VIs to count the number of pulses in a signal. You can use the Counter VIs to count up or count down with the different parts of a signal. You also can use the Counter VIs with gear tooth sensors to determine the speed of a rotating component such as a wheel on a robot.

You can configure counters to detect rising or falling edges. You also can calculate the frequency of a signal by measuring the period of time that elapses between edges.

Use the Open VI to create a **CounterDevRef** reference cluster that you then can wire to the other Counter VIs.

The following figure demonstrates how to return the count and direction of a counter.

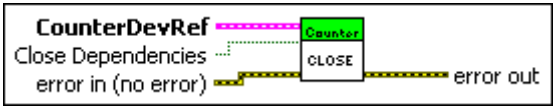


This example VI uses the ToDigitalSource VI to create the digital up and down sources for the counter. In this example, both digital sources are digital inputs. This example VI then opens a reference to and starts a counter. If the counter is stopped, the Get VI returns a **Stopped** value of TRUE, and the TRUE case of the Case structure executes. The counter stops counting, and this example VI closes the counter reference. If the counter is not stopped, the Get VI returns a **Stopped** value of FALSE, and

the FALSE case of the Case structure executes. In this case, the example VI continuously returns the count and direction of the counter until you press the **Stop** button on the front panel.

Close.vi

Closes the reference to the counter you specify. Use this VI to close each counter reference that you open with the Open VI. If you do not stop a running counter with the Stop VI, closing the reference to the counter stops the counter.



CounterDevRef specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

Close Dependencies, when TRUE, closes any digital inputs or analog trigger outputs used by the counter when you close the reference to the counter.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

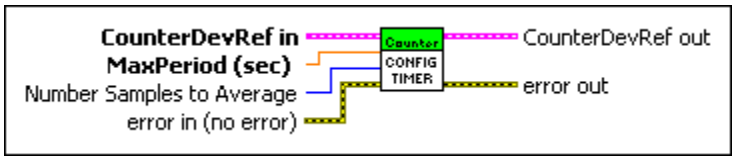
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

ConfigureTimer.vi

Configures the timer on the counter. Use this VI to specify the maximum time between pulses that can elapse before the application considers the counter stopped and to specify the number of samples of the timer to average when calculating the period. Use the Get VI to determine whether the counter is stopped.



CounterDevRef in specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

MaxPeriod (sec) specifies the maximum time that can elapse between pulses before the CompactRIO device considers the counter stopped. The default is 0.05 seconds.

Number Samples to Average specifies the number of samples of the timer to average when calculating the period. Perform averaging to account for mechanical imperfections. You can specify a value between 1 and 128. The default is 1.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CounterDevRef out returns a reference to the counter.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between **`Ctr 0`** and **`Ctr 7`**. If **CntIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

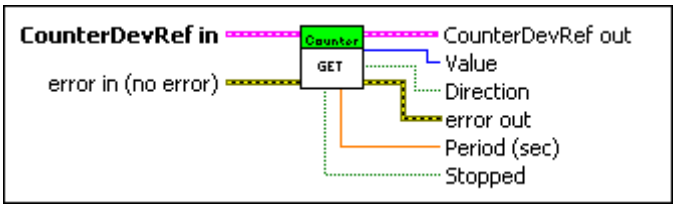
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get.vi

Returns information, such as number of counts and the period, from the counter you specify.

The Get VI returns the accumulated count from the counter unless you reset the counter with the Reset VI. When the value of the counter exceeds the maximum value of **Value**, the counter resets to zero automatically and continues counting. The maximum value of the counter is 4,294,967,295.



CounterDevRef in specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CounterDevRef out returns a reference to the counter.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CntIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

Value returns the count from the counter.

Direction returns TRUE if the counter last moved in a direction corresponding to an increase in the count. **Direction** returns FALSE if the counter last moved in a direction corresponding to a decrease in the count.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

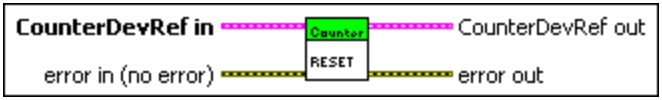
source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Period (sec) returns the average period, in seconds, between pulses of the counter. If you know the linear or rotational distance that the robot moved between pulses, you can use the period to determine linear or rotational velocity.

Stopped returns TRUE if the counter is stopped. The CompactRIO device considers a counter stopped if the CompactRIO device does not detect a new pulse and the number of counts does not change during the period you specify with the VI.

Reset.vi

Resets the counter you specify to 0. Use the Stop VI if you want to stop the counter.



CounterDevRef in specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CounterDevRef out returns a reference to the counter.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CntIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

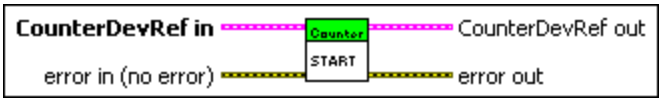
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Start.vi

Starts the counter that you specify. Use the Open VI to open a reference to the counter before you use this VI. Use the Reset VI if you want to reset the counter.



CounterDevRef in specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CounterDevRef out returns a reference to the counter.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CntIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

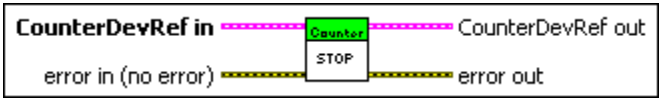
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Stop.vi

Stops the counter that you specify but does not reset the counter to zero. Use the Reset VI if you want to reset the counter without stopping the counter.



CounterDevRef in specifies a reference to the counter you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

CounterDevRef out returns a reference to the counter.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CntIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

20

DigitalInput VIs

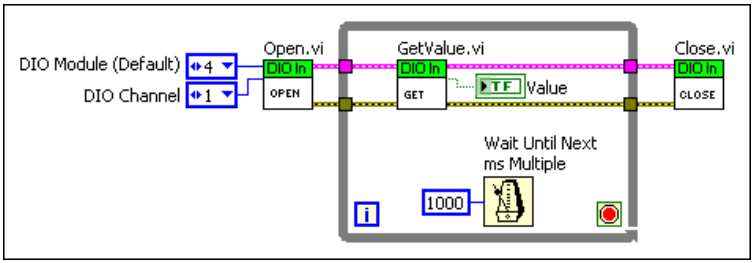
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the DigitalInput VIs.

Use the DigitalInput VIs to return digital input values, such as values that sensors send to the CompactRIO device.

You can connect a digital line to either a digital sidecar or directly to a digital module on the CompactRIO device. A digital line can accept input signals or send output signals. Digital inputs accept signals from sensors, such as limit switches or touch sensors, whose signal has a value of either 0 or 1.

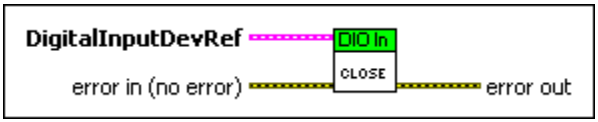
Use the Open VI to create a **DigitalInputDevRef** reference cluster that you then can wire to the other DigitalInput VIs.

The following figure demonstrates how to read and display the value of a digital sensor, such as a limit switch or a touch sensor, every second. The digital line is connected to channel 1 of the digital sidecar, which in turn is connected to the digital module in slot 4 of the CompactRIO device.



Close.vi

Closes the reference to the digital input you specify. Use this VI to close each digital input reference that you open with the Open VI.



DigitalInputDevRef specifies a reference to the digital input you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

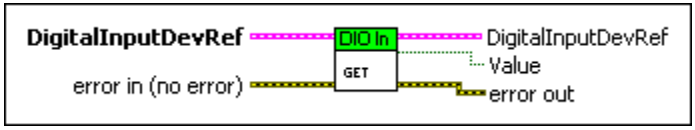
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetValue.vi

Returns the value of the digital input that you specify.



DigitalInputDevRef specifies a reference to the digital input you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalInputDevRef returns a reference to the digital input.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of **`Slot 4`** or **`Slot 6`**. If **DIO Module** returns a value of **`Default`**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between **`DIO 1`** and **`DIO 14`**. If **DIO Channel** returns a value of **`Invalid`**, the VI did

not find the digital channel you specified, and this VI returns an error.

Value returns the Boolean value of the digital input.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

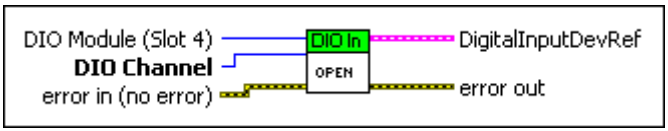
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Creates a reference to the digital input you specify. You must open a reference before using any other VIs on this palette.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. Select a value between **DIO 1** and **DIO 14**. The default is **DIO 1**. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalInputDevRef returns a reference to the digital input.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of **`Slot 4`** or **`Slot 6`**. If **DIO Module** returns a value of **`Default`**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between `DIO 1` and `DIO 14`. If **DIO Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

21

DigitalOutput VIs

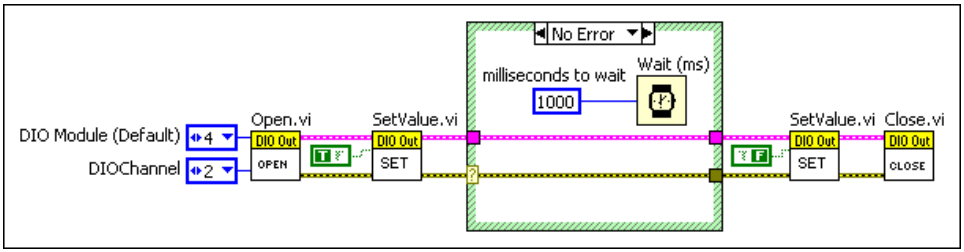
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the DigitalOutput VIs.

Use the DigitalOutput VIs to open or close a reference to a digital output and to set the value of the digital output.

You can connect a digital line to either a digital sidecar or directly to a digital module on the CompactRIO device. A digital line can accept input signals or send output signals. Digital outputs can emit digital pulses. You can use digital outputs to activate a sensor or turn on an LED.

Use the Open VI to create a **DigitalOutputDevRef** reference cluster that you then can wire to the other DigitalOutput VIs.

The following figure demonstrates how to emit a pulse on a digital line for 1000 milliseconds, or one second. The digital line is connected to channel 2 of the digital sidecar, which in turn is connected to the digital module in slot 4 of the CompactRIO device.

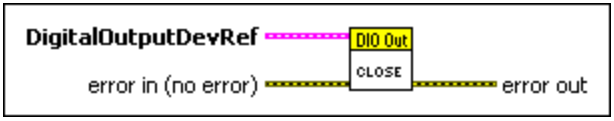


In the previous figure, the SetValue VI sets the digital output to TRUE for the duration of the pulse, which the Wait (ms) function specifies. The SetValue VI then sets the digital output back to FALSE at the end of the pulse. In this example, the pulse is software-timed. Alternatively, you can use the Pulse VI and hardware timing to specify the duration of the pulse.

Use software timing to emit pulses over longer durations. The Timing functions have a resolution of milliseconds. For example, you can use the digital output to turn on an LED for several seconds and then turn it off.

Close.vi

Closes the reference to the digital output you specify. Use this VI to close each digital output reference that you open with the Open VI.



error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalOutputDevRef specifies a reference to the digital output you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

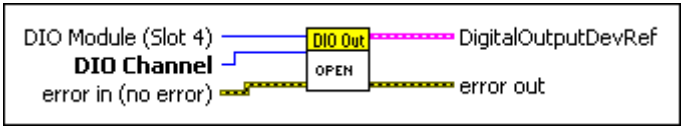
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Creates a reference to the digital output you specify. You must open a reference before using any other VIs on this palette.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital

module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. Select a value between **DIO 1** and **DIO 14**. The default is **DIO 1**. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalOutputDevRef returns a reference to the digital output.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of **`Slot 4`** or **`Slot 6`**. If **DIO Module** returns a value of **`Default`**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between **`DIO 1`** and **`DIO 14`**. If **DIO Channel** returns a value of **`Invalid`**, the VI did not find the digital channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

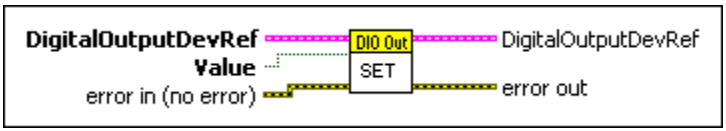
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetValue.vi

Specifies the value of the digital output you specify. You can use this VI to specify the starting value of the digital output before you use the Pulse VI.



Value specifies the Boolean value of the digital output.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs

while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalOutputDevRef specifies a reference to the digital output you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

DigitalOutputDevRef returns a reference to the digital output.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between `DIO 1` and `DIO 14`. If **DIO Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

22

Digital Output Advanced VIs

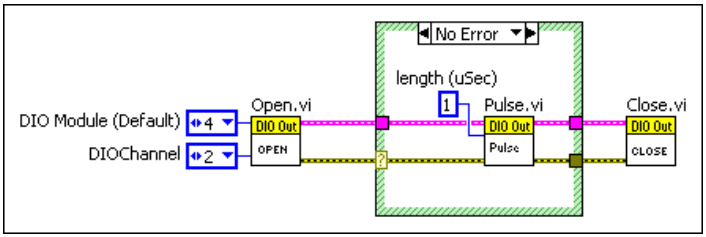
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Digital Output Advanced VIs.

Use the Digital Output Advanced VIs to emit digital pulses from the CompactRIO device.

You can connect a digital line to either the digital sidecar or to a digital module on the CompactRIO device. A digital line can accept input signals or send output signals. Digital outputs can emit digital pulses. You can use digital outputs to activate a sensor or turn on an LED.

Use the DigitalOutput Open VI to create a **DigitalOutputDevRef** reference cluster that you then can wire to the Digital Output Advanced VIs.

The following figure demonstrates how to emit a pulse on a digital line for one microsecond. The digital line is connected to channel 2 of the digital sidecar, which in turn is connected to the digital module in slot 4 of the CompactRIO device.

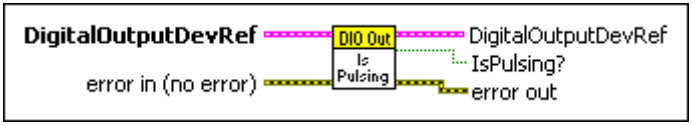


In the previous figure, the Pulse VI sets the digital output to the opposite of the current value for the duration you specify with the **length (uSec)** input. The Pulse VI emits a hardware-timed pulse. Alternatively, you can use the SetValue VI and software timing to specify the duration of the pulse.

Use hardware timing to emit a pulse of a very short duration. You can set the **length (uSec)** input of the Pulse VI to a value between 0 and 255 microseconds. For example, you can use the digital output to emit a short ping from an ultrasonic sensor.

IsPulsing.vi

Returns whether the digital output is emitting a pulse.



error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalOutputDevRef specifies a reference to the digital output you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

DigitalOutputDevRef returns a reference to the digital output.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between `DIO 1` and `DIO 14`. If **DIO Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

IsPulsing? returns TRUE if the digital output is emitting a hardware-timed pulse. Otherwise, **IsPulsing?** returns FALSE.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

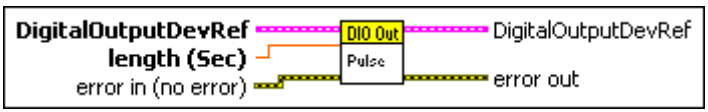
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Pulse.vi

Emits a hardware-timed pulse for a duration that you specify. For example, you can use this VI to emit a ping from an ultrasonic sensor. This VI sets the value of the digital output to the opposite of the current value. You can use the SetValue VI to specify the starting value of the digital output before you use the Pulse VI.



length (Sec) specifies the duration, in seconds, of the pulse the digital output emits. You can specify a value between 7 E-6 and 1.66 E-3. The default is 1.5 E-5.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DigitalOutputDevRef specifies a reference to the digital output you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

DigitalOutputDevRef returns a reference to the digital output.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel returns the channel of the **DIO Module** that you want to use. **DIO Channel** can return a value between `DIO 1` and `DIO 14`. If **DIO Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

23

DriverStation VIs

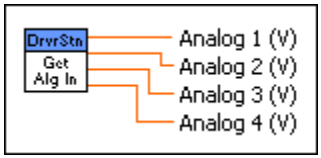
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the DriverStation VIs.

Use the DriverStation VIs to send data between the robot, driver station, and host computer. The DriverStation VIs can handle both analog and digital data.

You can use the DriverStation VIs to handle input data from sensors connected to the driver station, to send data from the driver station to LEDs or other sensors, and to send information from the robot to the host computer.

Get Analog Input.vi

Returns the voltage of each of the analog inputs on the driver station.



Analog 1 (V) returns the voltage of the first analog input on the driver station.

Analog 2 (V) returns the voltage of the second analog input on the driver station.

Analog 3 (V) returns the voltage of the third analog input on the driver station.

Analog 4 (V) returns the voltage of the fourth analog input on the driver station.

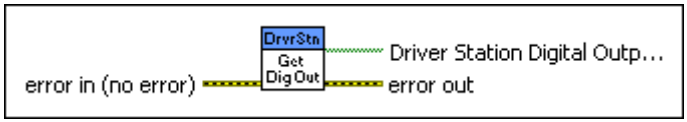
Get Digital Input.vi

Returns the value of each of the digital inputs on the driver station.

Driver Station Digital Input (1-8) returns the value of each of the digital inputs on the driver station.

Get Digital Output.vi

Returns the value of each of the digital outputs on the driver station.



error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Driver Station Digital Output (1-8) returns the value of each of the digital outputs on the driver station.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

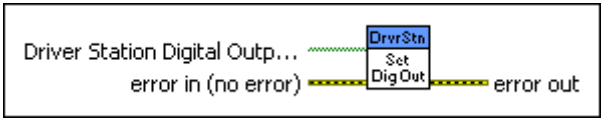
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set Digital Output.vi

Specifies the value of each of the digital outputs on the driver station. The UpdateDashboard VI must be running before you can use this VI.



Driver Station Digital Output (1-8) specifies the value of each of the digital outputs on the driver station.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set User Data.vi

Specifies the user data to send from the host computer to the driver station.

User Data (984 byte) specifies the user data to send from the host computer to the driver station.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

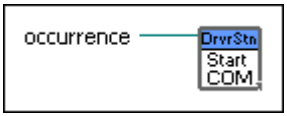
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Start Communication.vi

Starts the communication loop between the CompactRIO device and the driver station. This VI also initializes the host computer, joystick, and driver station data caches, as well as the system watchdog on the CompactRIO device.

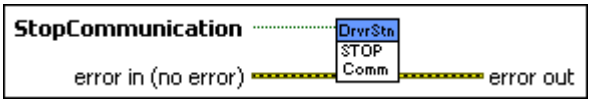
This VI runs continuously and regularly checks for information from the host computer, joystick, and driver station.



occurrence pauses the operation of the Robot Main VI in the FRC robot project that you deploy to and run on the CompactRIO device. Operation of the Robot Main VI resumes after the Start Communication VI data caches update. The **occurrence** reference comes from the Create Occurrence VI.

Stop Communication.vi

Stops the communication loop between the CompactRIO device and the driver station. This VI also closes the host computer, joystick, and driver station data caches.



StopCommunication, when TRUE, stops the communication loop between the CompactRIO device and the driver station and closes the host computer, joystick, and driver station data caches. When FALSE, **StopCommunication** has no effect on the communication loop between the CompactRIO device and the driver station or the host computer, joystick, and driver station data caches.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

24

Encoder VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Encoder VIs.

Use the Encoder VIs to get information about the rotation of a gear or motor, as measured by a quadrature encoder.

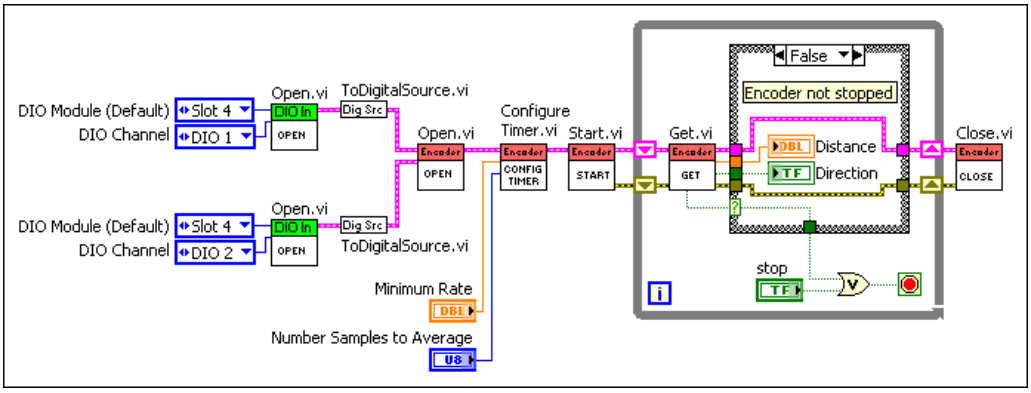
A quadrature encoder can count the number of rotations as well as determine the direction of rotation. Quadrature encoders use two digital sources that are out of phase to determine the count and direction of rotation. Timing is critical with encoder measurements. A 128-count encoder returns 128 pulses per encoder revolution. If you connect this encoder directly to a wheel with a six-inch diameter, the circumference of the wheel is 6π inches, and the resolution of the encoder is approximately $6\pi/128 = 0.15$ inches.

The FPGA on the CompactRIO device contains four quadrature decoders that perform 4x decoding and eight decoders that perform 1x and 2x decoding to interpret the information the encoders return. The decoders are not tied to a specific module or channel on the CompactRIO device. You can reserve and release decoders as needed.

The Encoder VIs update the count for each rising and falling edge the encoder generates. You can use the Encoder VIs to configure the decoders to increment the count for forward rotations and decrement the count for reverse rotations, or vice versa.

Use the Open VI to create an **EncoderDevRef** reference cluster that you then can wire to the other Encoder VIs. Use the ToDigitalSource VI to create a digital source that you can wire to the **ASource** and **BSource** inputs of the Open VI.

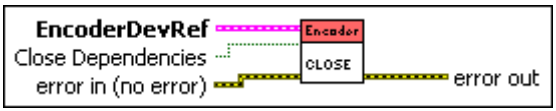
The following figure demonstrates how to return the count and direction of an encoder.



This example VI uses the ToDigitalSource VI to create the digital sources for the encoder. In this example, both digital sources are digital inputs. This example VI then opens a reference to an encoder, specifies a minimum pulse rate for determining if the encoder is stopped, and starts decoding the encoder signals. If the encoder is stopped, the Get VI returns a **Stopped** value of TRUE, and the TRUE case of the Case structure executes. The encoder stops decoding, and this example VI closes the encoder reference. If the encoder is not stopped, the Get VI returns a **Stopped** value of FALSE, and the FALSE case of the Case structure executes. In this case, the example VI continuously returns the distance moved and direction of the encoder until you press the **Stop** button on the front panel.

Close.vi

Closes the reference to the encoder you specify. Use this VI to close each encoder reference that you open with the Close VI. If you do not stop a running encoder with the Stop VI, closing the reference to the encoder stops the encoder.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

Close Dependencies specifies to close any digital inputs when you close the reference to the encoder. When TRUE, **Close Dependencies** closes the digital inputs.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

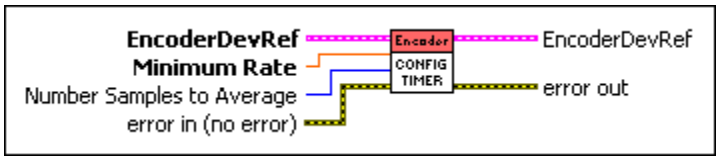
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

ConfigureTimer.vi

Configures the timer for the encoder you specify. Use this VI to specify the minimum rate at which the CompactRIO device must receive pulses from the encoder before the device considers the encoder stopped. You also can use this VI to specify the number of samples of the timer to average when calculating the pulse rate. Use the Get VI to determine whether the encoder is stopped.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

Minimum Rate specifies the minimum rate, in pulses per second, at which the CompactRIO device must receive pulses from the encoder before the device considers the encoder stopped. The default is 20. Use the VI to return the stopped status of the encoder.

Number Samples to Average specifies the number of samples of the timer to average when calculating the pulse rate. Perform averaging to account for mechanical imperfections. You can specify a value between 1 and 128. The default is 1.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between **`Enc 0`** and **`Enc 3`**. If **EncoderIndex** returns a value of **`Invalid`**, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between **`Ctr 0`** and **`Ctr 7`**. If **CounterIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

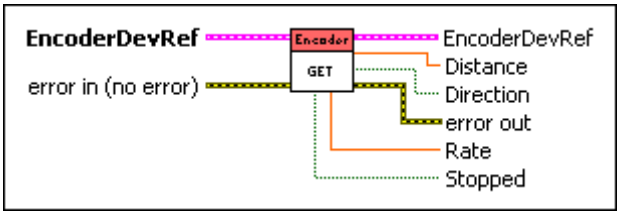
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get.vi

Returns information about the count, direction, and period of the encoder you specify, as well as whether the encoder is stopped.

This VI returns the count of the encoder you specify but does not reset the count of the encoder. If you call this VI repeatedly, this VI returns an accumulated count. Consider when the encoder was last reset when using this VI.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between `Enc 0` and `Enc 3`. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CounterIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

Distance returns the distance, in inches, that the wheel has traveled. **Distance** takes the count from the FPGA and divides it by 1, 2, or 4, depending on the corresponding encoder type. That value is then multiplied by the value of the **DistancePerCount** parameter from the VI, resulting in the **Distance** value.

Direction returns TRUE if the encoder last moved in a direction corresponding to an increase in the count. **Direction** returns FALSE if the encoder last moved in a direction corresponding a decrease in the count.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

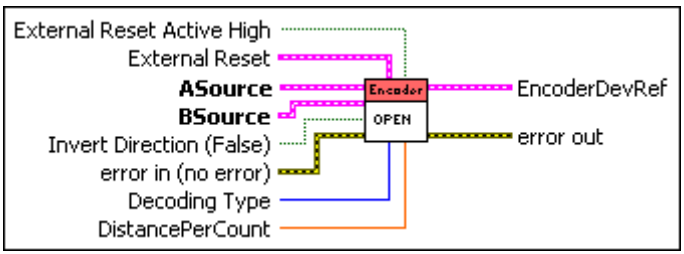
Rate returns the rate, in Hz, of pulses of the encoder.

Stopped returns TRUE if the encoder is stopped. The CompactRIO device considers an encoder stopped if the rate at which the CompactRIO device detects pulses from the encoder is less than the minimum rate you specify with the VI.

Open.vi

Opens a reference to the encoder you specify. You must open a reference before using any other VIs on this palette.

Use the ToDigitalSource VI to create a digital source that you can wire to the **ASource** and **BSource** inputs of this VI. After you open a reference to the encoder, you can use the Start and Stop VIs to start and stop the encoder, respectively.



External Reset Active High specifies whether the encoder resets when the External Reset digital source is high or low. When **External Reset Active High** is FALSE, the encoder resets when the digital source is low. When **External Reset Active High** is TRUE, the encoder resets when the digital source is high. The default is TRUE.

External Reset configures the digital source for the external signal that resets the encoder.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerMode? specifies, when TRUE, to use an analog trigger output instead of a digital input as the digital source.

DigitalMode specifies information about the digital input you want to use as the digital source.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Disabled**, the FPGA on the CompactRIO device disables the routing destination.

AnalogTriggerMode specifies information about the analog trigger output you want to use as the digital source.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

OutputType specifies the output of the analog trigger that you want to use as the digital source.

ASource specifies information about the first digital source of the encoder. You can use either a digital input or an analog trigger output as the digital source. Use the VI to create this digital source.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerMode? specifies, when TRUE, to use an analog trigger output instead of a digital input as the digital source.

DigitalMode specifies information about the digital input you want to use as the digital source.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Disabled**, the FPGA on the CompactRIO device disables the routing destination.

AnalogTriggerMode specifies information about the analog trigger output you want to use as the digital source.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

OutputType specifies the output of the analog trigger that you want to use as the digital source.

BSource specifies information about the second digital source of the encoder. You can use either a digital input or an analog trigger output as the digital source. Use the VI to create this digital source.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerMode? specifies, when TRUE, to use an analog trigger output instead of a digital input as the digital source.

DigitalMode specifies information about the digital input you want to use as the digital source.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Disabled**, the FPGA on the CompactRIO device disables the routing destination.

AnalogTriggerMode specifies information about the analog trigger output you want to use as the digital source.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

OutputType specifies the output of the analog trigger that you want to use as the digital source.

Invert Direction (False) specifies, when TRUE, that the encoder increases the count when the **ASource** signal leads the **BSource** signal and decreases the count when the **BSource** signal leads the **ASource** signal. **Invert Direction (False)** specifies, when FALSE, that the encoder increases the count when the **BSource** signal leads the **ASource** signal and decreases the count when the **ASource** signal leads the **BSource** signal. The default is FALSE.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between `Enc 0` and `Enc 3`. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CounterIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

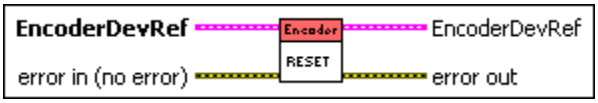
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Reset.vi

Resets the count of the encoder you specify to 0. Use the Stop VI if want the encoder to continue counting but want to stop decoding the signals the encoder returns.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between `Enc 0` and `Enc 3`. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CounterIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

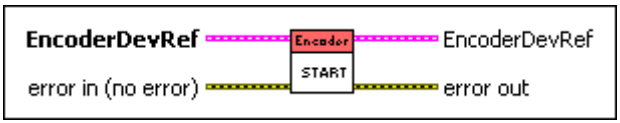
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Start.vi

Starts decoding the signals that the encoder you specify returns. Use the Open VI to open a reference to the encoder before you use this VI. Use the Reset VI if you want to reset the count of the encoder.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between `Enc 0` and `Enc 3`. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CounterIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

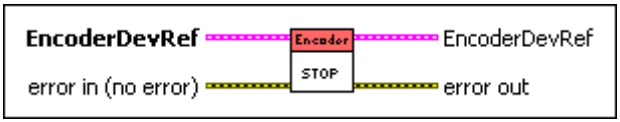
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Stop.vi

Stops decoding the signals the encoder you specify returns. Use the Reset VI if you want to reset the count of the encoder.



EncoderDevRef specifies a reference to the encoder you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderIndex specifies the index of the reserved encoder. **EncoderIndex** can specify a value between **Enc 0** and **Enc 3**. If **EncoderIndex** is **Invalid**, this VI returns an error.

CounterIndex specifies the index of the reserved counter. **CounterIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CounterIndex** is **Invalid**, this VI returns an error.

DistancePerCount scales the pulses of the encoder into engineering units for the **Distance** parameter of the VI. For example, if the wheel diameter is six inches and the encoder pulses 64 times per rotation of the wheel, then a scaling factor of 6/64 = .2945 returns the number of inches the wheel has turned. In this case, you input .2945

Decoding Type specifies the type of decoding used to decode the quadrature signal. **Decoding Type** supports three types of decoding for quadrature encoders: 1x, 2x, and 4x.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

EncoderDevRef returns a reference to the encoder.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between `Enc 0` and `Enc 3`. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder you specified, and this VI returns an error.

CounterIndex returns the index of the reserved counter. **CounterIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CounterIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

DistancePerCount returns the pulses of the encoder as engineering units.

Decoding Type returns the type of decoding used to decode the quadrature signal.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

25

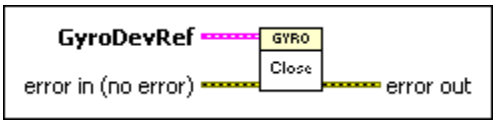
Gyro VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Gyro VIs.

Use the Gyro VIs to set and monitor the heading, or direction, of the robot, as measured by a gyroscope transducer. The Gyro VIs measure the instantaneous rate of rotation, or yaw rate, of the robot. The gyroscope driver runs in the background, constantly summing the rate information to determine heading. Use the Reset VI to initialize gyroscope measurements while the robot is stationary to determine the offset or bias value. The bias value is the point of origin, set to zero, with no rotation, that is subtracted from subsequent heading samples. Negative heading values indicate rotation to the left, positive values indicate rotation to the right. You also can use the Gyro VIs to measure how many degrees off of an original heading a robot has turned and to monitor the sensitivity in volts per degree per second that the robot turns.

Close.vi

Closes the reference to the gyroscope you specify. Use this VI to close each gyroscope reference that you open with the Open VI.



GyroDevRef specifies a reference to the gyroscope you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Offset specifies the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

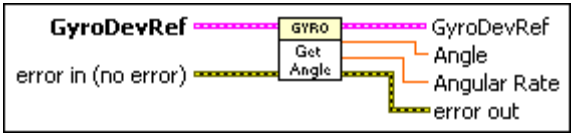
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetAngle.vi

Returns the angle, in degrees, from the original heading that the robot is currently facing.



GyroDevRef specifies a reference to the gyroscope you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Offset specifies the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

GyroDevRef returns the status of the gyroscope.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Offset returns the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) returns the voltage per degree of rotation per second. One of the parameters used by the VI to calculate and report the heading of the robot.

Angle returns the angle, in degrees, that the robot is currently facing.

Angular Rate scales the angular rate voltage signal from the gyroscope into engineering units of degrees per second. The accumulator on the FPGA numerically integrates this value to return the **Angle** of the gyroscope.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

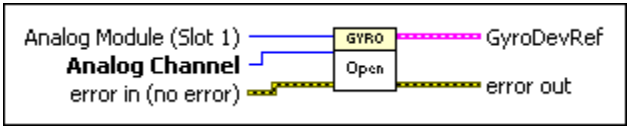
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the gyroscope you specify. You must open a reference before using any other VIs on this palette.

Note Gyroscopes use accumulators, which you can access only through the analog module in slot 1 of the CompactRIO device. You therefore must set **Analog Module** to **Slot 1**.



Analog Module (Slot 1) specifies the slot number on the CompactRIO device of the analog module you want to use. You must select **Slot 1**.

Analog Channel specifies the channel of the **Analog Module** you want to use. Select **AI 1** or **AI 2**. The default is **AI 1**. If **Analog Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

GyroDevRef returns the status of the gyroscope.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Offset returns the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) returns the voltage per degree of rotation per second. One of the parameters used by the VI to calculate and report the heading of the robot.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Reset.vi

Resets the gyroscope to a heading of zero. Use this VI to recalibrate the gyroscope after significant drift.

GyroDevRef specifies a reference to the gyroscope you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Offset specifies the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

GyroDevRef returns the status of the gyroscope.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of **`Slot 1`** or **`Slot 2`**.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between **`AI 1`** and **`AI 8`**. If **Analog Channel** returns a value of **`Invalid`**, the VI did not find the analog channel you specified, and this VI returns an error.

Offset returns the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) returns the voltage per degree of rotation per second. One of the parameters used by the VI to calculate and report the heading of the robot.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

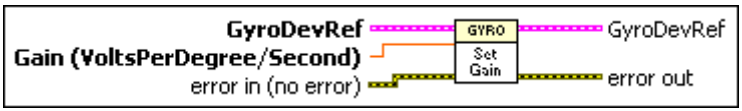
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetGain.vi

Specifies the voltage used per degree of rotation per second. This VI specifies one of the parameters used by the GetAngle VI to calculate and report the heading of the robot.



GyroDevRef specifies a reference to the gyroscope you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Analog Module specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **Slot 1** or **Slot 2**.

Analog Channel specifies the channel of the **Analog Module** you want to use. **Analog Channel** can specify a value between **AI 1** and **AI 8**. If **Analog Channel** is **Invalid**, this VI returns an error.

Offset specifies the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

Gain (VoltsPerDegree/Second) specifies the voltage per degree of rotation per second. This parameter is used by the VI to calculate and report the heading of the robot.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

GyroDevRef returns the status of the gyroscope.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Analog Module returns the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can return a value of `Slot 1` or `Slot 2`.

Analog Channel returns the channel of the **Analog Module** you want to use. **Analog Channel** can return a value between `AI 1` and `AI 8`. If **Analog Channel** returns a value of `Invalid`, the VI did not find the analog channel you specified, and this VI returns an error.

Offset returns the difference between the expected 0 heading value and the reported 0 heading value before the robot starts moving.

Gain (VoltsPerDegree/Second) returns the voltage per degree of rotation per second. One of the parameters used by the VI to calculate and report the heading of the robot.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

26

I2C VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the I2C VIs.

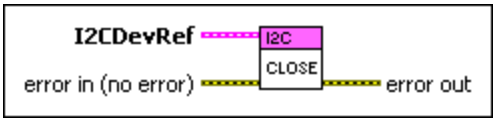
Use the Inter-Integrated Circuit (I2C) VIs to access and monitor sensors connected to an I2C communications bus.

Each digital module on the CompactRIO device has one independent I2C communications bus. Each sensor connected to an I2C communications bus must have a unique device address. Some sensors, such as the Devantech SRF08 ultrasonic sensor, have configurable addresses. Others, such as the HiTechnic NXT sensors, do not have configurable addresses.

Use the Open VI to create an **I2CDevRef** reference cluster that you then can wire to the other I2C VIs.

Close.vi

Closes a reference to the I2C sensor you specify. Use this VI to close each I2C sensor reference that you open with the Open VI.



I2CDevRef specifies a reference to the I2C sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address specifies the unique device address of the I2C sensor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

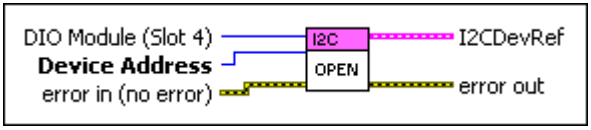
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the I2C sensor you specify. You must open a reference before using any other VIs on this palette. After you open a reference to an I2C sensor, you can use the other I2C VIs to access and monitor the sensor.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address specifies the unique device address of the I2C sensor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

I2CDevRef returns a reference to the I2C sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address returns the unique device address of the I2C sensor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

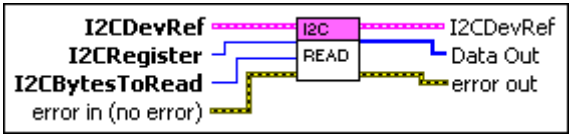
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Read.vi

Reads between one and four bytes in a single transaction from the I2C sensor you specify.



I2CDevRef specifies a reference to the I2C sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address specifies the unique device address of the I2C sensor.

I2CRegister specifies the register of the I2C sensor you want to read.

I2CBytesToRead specifies the number of bytes to read from the **I2CRegister** of the I2C sensor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs

while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

I2CDevRef returns a reference to the I2C sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address returns the unique device address of the I2C sensor.

Data Out returns the array of bytes this VI read from the I2C sensor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

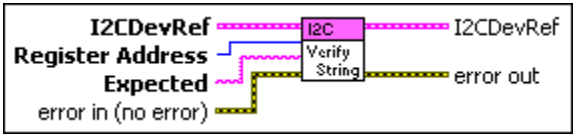
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

VerifyString.vi

Reads consecutive registers from the I2C sensor you specify and compares the register strings to an expected value. This VI returns an error if the register strings do not match.



I2CDevRef specifies a reference to the I2C sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address specifies the unique device address of the I2C sensor.

Register Address specifies the starting register to read from the I2C sensor.

Expected specifies the string that the I2C sensor is expected to return. This VI returns an error if the I2C sensor does not return this string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

I2CDevRef returns a reference to the I2C sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address returns the unique device address of the I2C sensor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

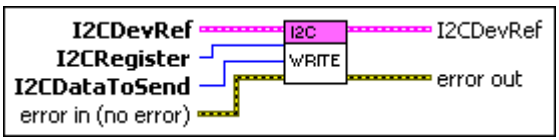
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Write.vi

Writes a byte to the I2C sensor you specify.



I2CDevRef specifies a reference to the I2C sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address specifies the unique device address of the I2C sensor.

I2CRegister specifies the register of the I2C sensor to which you want to write.

I2CDataToSend specifies the byte you want to write to the I2C sensor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

I2CDevRef returns a reference to the I2C sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Device Address returns the unique device address of the I2C sensor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

27

Interrupts VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Interrupts VIs.

Use the Interrupts VIs to configure interrupts for the CompactRIO device. You can program interrupt handlers to perform certain tasks when the CompactRIO device receives an interrupt.

Use the Open VI to create an **InterruptRef** reference cluster that you then can wire to the other Interrupts VIs.

Close.vi

Closes the reference to the interrupt you specify. Use this VI to close each interrupt reference that you open with the Open VI.

InterruptRef specifies a reference to the interrupt you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

InterruptIndex specifies the index of the reserved interrupt. **InterruptIndex** can specify a value between 0 and 7. If **InterruptIndex** is **Invalid**, this VI returns an error.

Close Dependencies specifies, when TRUE, to close any digital inputs or analog trigger outputs that the interrupt uses when you close the reference to the interrupt.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

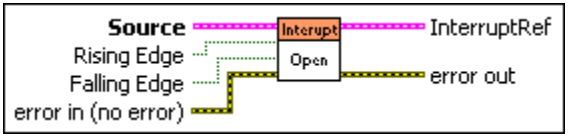
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the interrupt you specify. You must open a reference before using any other VIs on this palette.



Source specifies the source signal for the interrupt.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

AnalogTriggerMode? specifies, when TRUE, to use an analog trigger output instead of a digital input as the digital source.

DigitalMode specifies information about the digital input you want to use as the digital source.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Disabled**, the FPGA on the CompactRIO device disables the routing destination.

AnalogTriggerMode specifies information about the analog trigger output you want to use as the digital source.

AnalogTriggerIndex specifies the index of the reserved analog trigger. **AnalogTriggerIndex** can specify a value between **Trig 0** and **Trig 7**. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

OutputType specifies the output of the analog trigger that you want to use as the digital source.

Rising Edge specifies, when TRUE, to use the rising edge of the **Source** signal to trigger an interrupt handler.

Falling Edge specifies, when TRUE, to use the falling edge of the **Source** signal to trigger an interrupt handler.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

InterruptRef returns a reference to the interrupt.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

InterruptIndex returns the index of the reserved interrupt. **InterruptIndex** can return a value between 0 and 7.
If **InterruptIndex** returns a value of `Invalid`, the VI did not find the interrupt you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Wait.vi

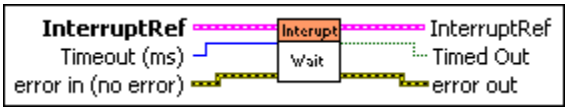
Specifies that the CompactRIO device waits for an interrupt. If the CompactRIO device receives an interrupt during the time you specify, this VI returns a **Timed Out** value of FALSE. You then can handle the interrupt as appropriate. If the CompactRIO device times out before receiving an interrupt, this VI returns a **Timed Out** value of TRUE. You can set this VI to wait for an interrupt indefinitely by setting **Timeout (ms)** to **–1**.

Note The use of interrupts is an advanced feature. Most teams might not need to use these VIs.

One common use case is to place this VI in an independent While Loop and set **Timeout (ms)** to **–1**. If the CompactRIO device receives an interrupt, this VI returns the **InterruptRef** and a **Timed Out** value of FALSE, and you handle the interrupt. You then return to the While Loop to wait indefinitely for another interrupt.

If you specify a program to handle the Stop derived state of the robot instead of aborting the robot program directly, do not set the Wait VI to wait for an interrupt indefinitely. Otherwise, the robot program might hang if no interrupts occur when you shut down the robot. Instead, specify a finite time for the CompactRIO device to wait for an interrupt. When the CompactRIO device times out, check if the robot is shutting down. If the robot is not shutting down, return to the Wait VI. Refer to the *Init, Execute, and Stop Derived States* section of Chapter 1, *Overview of the FIRST Robotics Competition*, for more information about the robot derived states.

Specifying a finite time to wait for an interrupt can introduce jitter into the interrupt handling process. The CompactRIO device might be handling a timeout at the same time an actual interrupt occurs. Therefore, the CompactRIO device takes longer to recognize and handle the interrupt. The smaller the **Timeout (ms)**, the greater the likelihood of introducing jitter.



InterruptRef specifies a reference to the interrupt you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

InterruptIndex specifies the index of the reserved interrupt. **InterruptIndex** can specify a value between 0 and 7. If **InterruptIndex** is **Invalid**, this VI returns an error.

Timeout (ms) specifies the time, in milliseconds, to wait for the interrupt. The default is –1, which specifies to wait indefinitely for an interrupt to occur.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

InterruptRef returns a reference to the interrupt.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

InterruptIndex returns the index of the reserved interrupt. **InterruptIndex** can return a value between 0 and 7.
If **InterruptIndex** returns a value of `Invalid`, the VI did not find the interrupt you specified, and this VI returns an error.

Timed Out returns TRUE if the time you specified with the **Timeout (ms)** input has passed without the CompactRIO device receiving an interrupt. Otherwise, **Timed Out** returns FALSE.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

28

IO VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the IO VIs.

Use the IO VIs to send and receive analog and digital data from a module on the CompactRIO device.

29

Joystick VIs

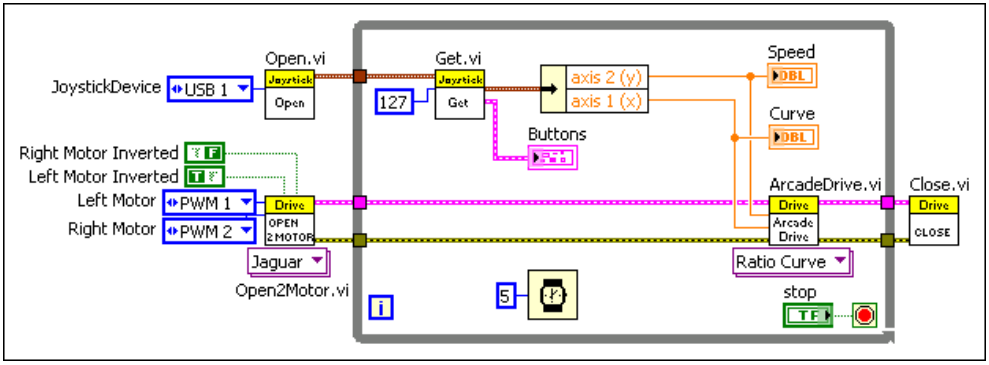
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Joystick VIs.

Use the Joystick VIs to get information about one or more joysticks you connect to the driver station. You can connect up to four joysticks to the driver station. In the Teleop mode of the *FIRST* Robotics Competition (FRC), the robot moves in response to commands it receives from the joysticks.

Use the Open VI to create a **JoystickDevRef** reference cluster that you then can wire to the other Joystick VIs.

Analog joysticks use a pair of potentiometers or Hall sensors and magnets to measure left-to-right movement and forward-and-back movement. The farther you move the joystick in a certain direction, the faster the object moves in that direction. Most analog joysticks have a resolution of 256 × 256. The Joystick VIs use axis values between –1.0 and 1.0, where 1.0 implies the fastest speed in one direction, and –1.0 implies the fastest speed in the opposite direction. Use the **Scaler** value of the Get and GetAxis VIs to scale the joystick values to values between –1.0 and 1.0. For example, if the resolution of the joystick is 256 × 256, set the **Scaler** value to 127. Values between –127 and 0 are scaled to values between –1.0 and 0.0, and values between 0 and 127 are scaled to values between 0.0 and 1.0.

The following figure demonstrates how to open a reference to joystick 1, get the button values and scaled axis values for that joystick, and use those values to specify the **Y axis value (Speed)** and **X axis value (Curve)** inputs for the ArcadeDrive VI. Joystick 1 has resolution of 256 × 256, so you set the **Scaler** input of the Get VI to 127.



The RobotDrive VIs in the previous figure control a two-wheeled robot using Jaguar motor controllers. The ArcadeDrive VI computes the motor speed from the **Y axis value (Speed)** and **X axis value (Curve)** values you specify.

Close.vi

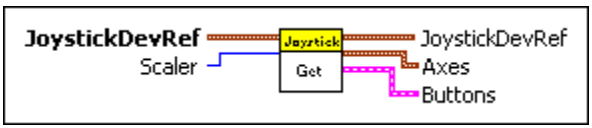
Closes a reference to the joystick you specify. Use this VI to close each joystick reference that you open with the Open VI.

JoystickDevRef specifies a reference to the joystick you want to use.

Device specifies the port on the driver station, from 1–4, to which you connected the joystick.

Get.vi

Returns button and scaled axis information about a joystick that you specify. Use the GetAxis VI to return the scaled value of a specific axis of the joystick. Use the GetRawValue VI to return button and raw axis information about the joystick.



JoystickDevRef specifies a reference to the joystick you want to use.

Device specifies the port on the driver station, from 1–4, to which you connected the joystick.

Scaler specifies the scaling value this VI uses to calculate the scaled joystick value. The default is 127.

JoystickDevRef returns a reference to the joystick.

Device returns the port on the driver station, from 1–4, to which you connected the joystick.

Axes returns the scaled axis information for the joystick.

axis 1 (x) returns the scaled x-axis value of the joystick that the *FIRST* Robotics Competition (FRC) kit provides. **axis 1 (x)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 2 (y) returns the scaled y-axis value of the joystick that the FRC kit provides. **axis 2 (y)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 3 (throttle) returns the scaled throttle value of the joystick that the FRC kit provides. **axis 3 (throttle)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 4 returns a scaled value of the joystick. The value to which **axis 4** corresponds is dependent on the joystick or axis configuration you use.

axis 5 returns a scaled value of the joystick. The value to which **axis 5** corresponds is dependent on the joystick or axis configuration you use.

axis 6 returns a scaled value of the joystick. The value to which **axis 6** corresponds is dependent on the joystick or axis configuration you use.

Buttons returns the values of the buttons on the joystick.

Button 1 returns TRUE when button 1 on the joystick is pressed.

Button 2 returns TRUE when button 2 on the joystick is pressed.

Button 3 returns TRUE when button 3 on the joystick is pressed.

Button 4 returns TRUE when button 4 on the joystick is pressed.

Button 5 returns TRUE when button 5 on the joystick is pressed.

Button 6 returns TRUE when button 6 on the joystick is pressed.

Button 7 returns TRUE when button 7 on the joystick is pressed.

Button 8 returns TRUE when button 8 on the joystick is pressed.

Button 9 returns TRUE when button 9 on the joystick is pressed.

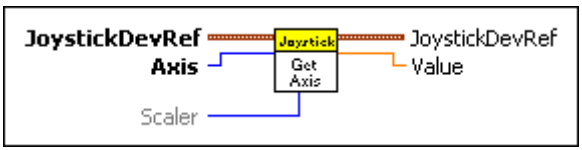
Button 10 returns TRUE when button 10 on the joystick is pressed.

Button 11 returns TRUE when button 11 on the joystick is pressed.

Button 12 returns TRUE when button 12 on the joystick is pressed.

GetAxis.vi

Returns the scaled value of an axis of a joystick that you specify. Use the GetRawValue VI to return raw axis information about the joystick.



JoystickDevRef specifies a reference to the joystick you want to use.

Device specifies the port on the driver station, from 1–4, to which you connected the joystick.

Axis specifies the axis whose scaled value you want this VI to return.

Scaler specifies the scaling value this VI uses to calculate the scaled joystick value. The default is 127.

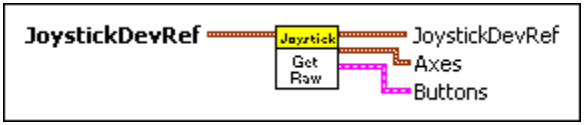
JoystickDevRef returns a reference to the joystick.

Device returns the port on the driver station, from 1–4, to which you connected the joystick.

Value returns the scaled axis value of the joystick you specified.

GetRawValue.vi

Returns button and raw axis information about a joystick that you specify. Use the Get VI to return button and scaled axis information about the joystick.



JoystickDevRef specifies a reference to the joystick you want to use.

Device specifies the port on the driver station, from 1–4, to which you connected the joystick.

JoystickDevRef returns a reference to the joystick.

Device returns the port on the driver station, from 1–4, to which you connected the joystick.

Axes returns the raw axis information for the joystick.

axis 1 (x) returns the raw x-axis value of the joystick that the *FIRST* Robotics Competition (FRC) kit provides. **axis 1 (x)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 2 (y) returns the raw y-axis value of the joystick that the FRC kit provides. **axis 2 (y)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 3 (throttle) returns the raw throttle value of the joystick that the FRC kit provides. **axis 3 (throttle)** might return the value of a different axis if you use a different joystick or configure the axes differently.

axis 4 returns a raw value of the joystick. The value to which **axis 4** corresponds is dependent on the joystick or axis configuration you use.

axis 5 returns a raw value of the joystick. The value to which **axis 5** corresponds is dependent on the joystick or axis configuration you use.

axis 6 returns a raw value of the joystick. The value to which **axis 6** corresponds is dependent on the joystick or axis configuration you use.

Buttons returns the values of the buttons on the joystick.

Button 1 returns TRUE when button 1 on the joystick is pressed.

Button 2 returns TRUE when button 2 on the joystick is pressed.

Button 3 returns TRUE when button 3 on the joystick is pressed.

Button 4 returns TRUE when button 4 on the joystick is pressed.

Button 5 returns TRUE when button 5 on the joystick is pressed.

Button 6 returns TRUE when button 6 on the joystick is pressed.

Button 7 returns TRUE when button 7 on the joystick is pressed.

Button 8 returns TRUE when button 8 on the joystick is pressed.

Button 9 returns TRUE when button 9 on the joystick is pressed.

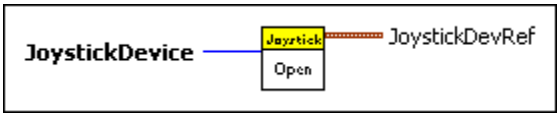
Button 10 returns TRUE when button 10 on the joystick is pressed.

Button 11 returns TRUE when button 11 on the joystick is pressed.

Button 12 returns TRUE when button 12 on the joystick is pressed.

Open.vi

Opens a reference to the joystick you specify. You must open a reference before using any other VIs on this palette. After you open a reference to the joystick, you can use the other Joystick VIs to read the button and axis values of the joystick.



JoystickDevice specifies the port on the driver station, from 1–4, to which you connected the joystick.

JoystickDevRef returns a reference to the joystick.

Device returns the port on the driver station, from 1–4, to which you connected the joystick.

30

MotorControl VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the MotorControl VIs.

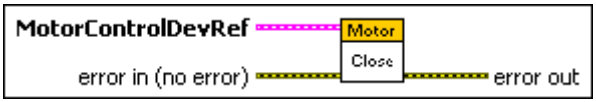
Use the MotorControl VIs to control the speed and direction of motors.

Use the Open VI to create a **MotorControlDevRef** reference cluster that you then can wire to the other MotorControl VIs.

Close.vi

Closes the reference to the motor controller you specify. Use this VI to close each motor control reference that you open with the Open VI.

The Close VI closes references to either the Jaguar or Victor motor control references that you open with the Open VI. For example, use the Jaguar instance of the Open VI to open a reference to the Jaguar motor controller and use the Close VI to close that reference.



MotorControlDevRef specifies a reference to the motor controller you want to use.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

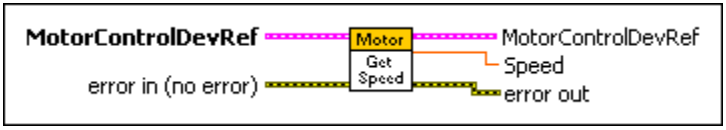
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetSpeed.vi

Returns the speed of the motor you specify.



MotorControlDevRef specifies a reference to the motor controller you want to use.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

MotorControlDevRef returns a reference to the motor controller you want to use.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Speed returns the scaled value from the PWM. The scaled value ranges from –1.0 to 1.0. Otherwise, the range of the scaled values depends on the custom scaling you specify with the **TransformRef** and **InvTransformRef** converters. Use these converters if you need to apply a non-linear transformation to the motor control signal to compensate for the characteristics of the motor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

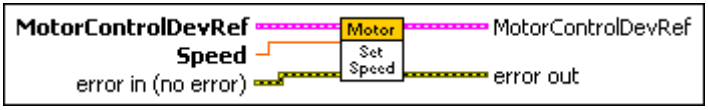
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetSpeed.vi

Specifies the motion of the PWM using a scaled value from –1.0 to 1.0. Use the SetValue VI to set the PWM using a raw value from 0 to 255.



MotorControlDevRef specifies a reference to the motor controller you want to use.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Speed specifies the scaled value of the PWM. If you use a Jaguar or Victor motor controller, the scaled value can range from –1.0 to 1.0. Otherwise, the range of the scaled values depends on the custom scaling you specify with the **TransformRef** and **InvTransformRef** converters.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

MotorControlDevRef returns a reference to the motor controller you want to use.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

31

PWM VIs

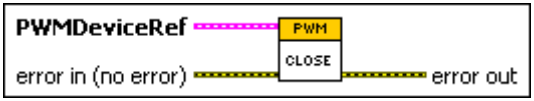
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the PWM VIs.

Use the PWM VIs to generate a hobby-style pulse width modulation (PWM) signal. You can use PWM signals to drive motor controllers. Use the VIs on this palette with custom motor controllers. Otherwise, use the VIs on the **RobotDrive** or **MotorControl** palettes to use PWM for those devices.

Use the Open VI to create a **PWMDeviceRef** reference cluster that you then can wire to the other PWM VIs.

Close.vi

Closes the reference to the PWM or motor controller you specify. Use this VI to close each PWM reference that you open with the Open VI.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the specific motor controller or servo Open VI to open this reference when possible. Otherwise, use the VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

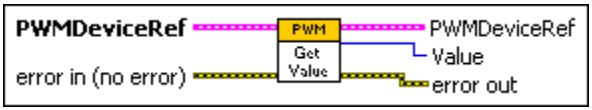
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetValue.vi

Returns the raw value from the PWM you specify. The raw PWM value ranges from 0 to 255. A value of 1 puts the motor controller in full reverse, a value of 255 puts the motor controller in full forward, and a value of 0 disables the motor controller. Use this VI with custom motor controllers. Use the GetSpeed VI on the MotorControl palette to return PWM values from the Jaguar or Victor motor controllers.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the specific motor controller or servo Open VI to open this reference when possible. Otherwise, use the VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Value returns the PWM value between 0 and 255. A value of 1 puts the motor controller in full reverse, a value of 255 puts the motor controller in full forward, and a value of 0 disables the motor controller.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

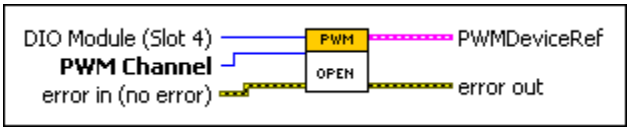
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the PWM port you specify. You must open a reference before using any other PWM VIs. After you open a reference to the PWM, you can use the Close VI to close the reference. The Open VI on the PWM palette opens references to custom motor controllers. You must use the Open VI on the RobotDrive or MotorControl palettes to open references to those devices. For example, use the Jaguar instance of the Open VI on the MotorControl palette to open a reference to the Jaguar motor controller.

By default, the center of the signal is 128.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. Select a value between **PWM 1** and **PWM 10**. The default is **PWM 1**. If **PWM Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

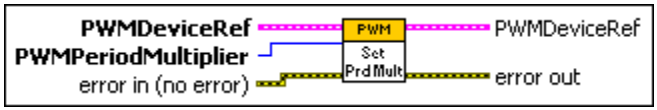
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetPeriodMultiplier.vi

Scales the period of the PWM you specify.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the specific motor controller or servo Open VI to open this reference when possible. Otherwise, use the VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

PWMPeriodMultiplier specifies how much to scale the period of the PWM. You can right-click the **PWMPeriodMultiplier** input and select **Create»Constant** to create a ring control from which you can select a multiplier. Otherwise, you can wire an unsigned 8-bit integer to the input according to the following table.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

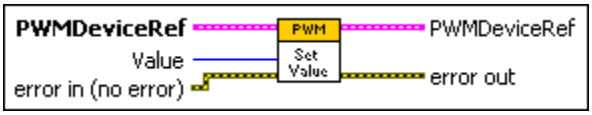
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetValue.vi

Specifies the motion of the PWM using a PWM value from 0 to 255. The raw PWM value ranges from 0 to 255. A value of 1 puts the motor controller in full reverse, a value of 255 puts the motor controller in full forward, and a value of 0 disables the motor controller. Use this VI with custom motor controllers. Use the SetSpeed VI on the MotorControl palette to specify PWM values for the Jaguar or Victor motor controllers.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the specific motor controller or servo Open VI to open this reference when possible. Otherwise, use the VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Value specifies the PWM value between 0 and 255. A value of 1 puts the motor controller in full reverse, a value of 255 puts the motor controller in full forward, and a value of 0 disables the motor controller.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

32

Relay VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Relay VIs.

Use the Relay VIs to manipulate relays. You can open and close relays to control the power source for a motor or other device.

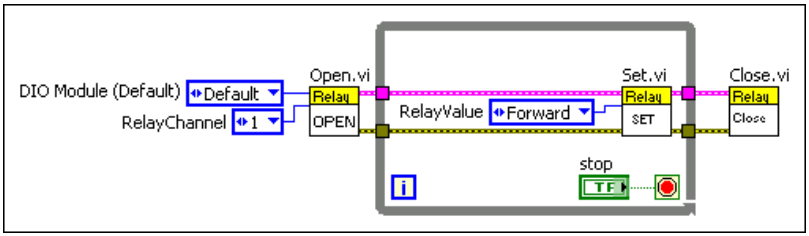
Use the Open VI to create a **RelayDevRef** reference cluster that you then can wire to the other Relay VIs.

Most sensors that you use in the *FIRST* Robotics Competition (FRC) can receive sufficient power from the CompactRIO device. However, motors require more power than the CompactRIO device can provide and therefore must receive power directly from a battery. You can use the CompactRIO device to operate a relay that connects a motor to a battery. When the relay is closed, current flows to the motor from the battery.

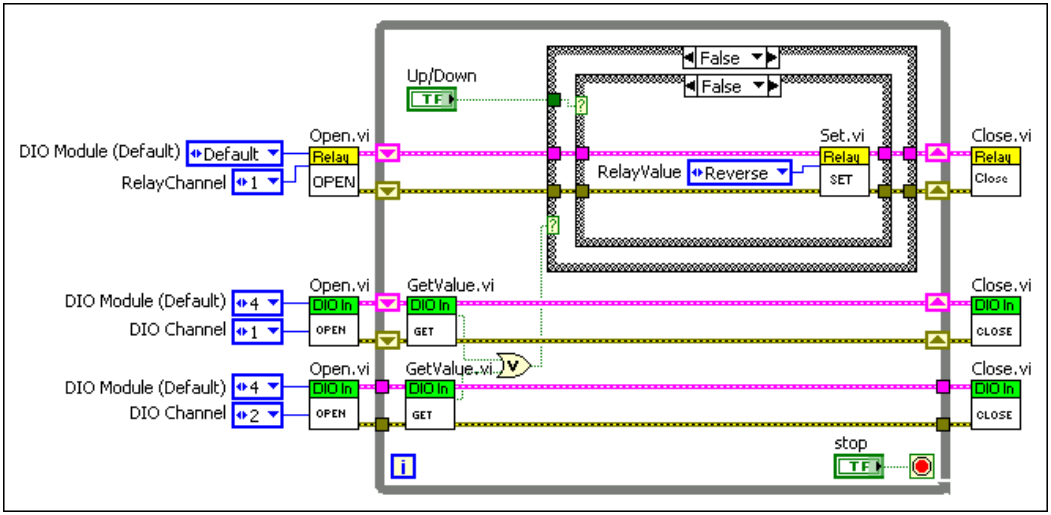
You also can use relays with a pressure switch and the Compressor VIs to maintain pressure with a compressor. When the pressure is low, the pressure switch can turn the relay on, which in turn can start the compressor.

Each digital module on the CompactRIO device contains eight relay channels, and each relay channel consists of two outputs. If you wire both outputs of a relay to the same motor, you can determine the direction, either forward or backward, in which the motor moves. If you wire the outputs to two different motors, you can specify only whether each motor does or does not move.

The following figure demonstrates how to set a relay such that the corresponding motor moves continuously in the forward direction. The relay is connected to relay channel 1 of the default digital module on the CompactRIO device.

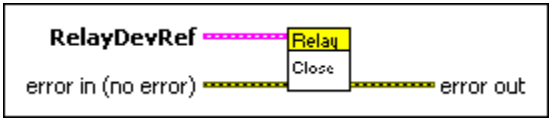


The following figure demonstrates how to use the Relay VIs and the DigitalInput VIs to control the direction of a motor and to stop the motor if either of two limit switches is triggered. A relay is connected to relay channel 1 of the default digital module on the CompactRIO device. Two digital input values, connected to channels 1 and 2, respectively, of the digital module in slot 4 of the CompactRIO device, are used as limit switches. If both limit switches are FALSE, the False case of the outer Case structure executes. The **Up/Down** Boolean then determines which case of the inner Case structure to execute. The True case moves the motor in the forward direction, and the False case moves the motor in the reverse direction. If one or both limit switches are TRUE, the True case of the outer Case structure executes, and the motor stops.



Close.vi

Closes the reference to the relay you specify. Use this VI to close each relay reference that you open with the Open VI.



RelayDevRef specifies a reference to the relay you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel specifies the channel of the **DIO Module** to which the relay is connected. **Relay Channel** can specify a value between **Relay 1** and **Relay 8**. If **Relay Channel** is **Invalid**, this VI returns an error.

Relay Direction specifies the direction in which a motor corresponding to the relay moves.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

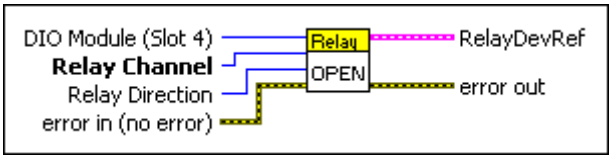
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the relay you specify. You must open a reference before using any other VIs on this palette.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel specifies the channel of the **DIO Module** to which the relay is connected. Select a value between **Relay 1** and **Relay 8**. The default is **Relay 1**. If **Relay Channel** is **Invalid**, this VI returns an error.

Relay Direction specifies the direction in which a motor corresponding to the relay moves.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

RelayDevRef returns a reference to the relay.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel returns the channel of the **DIO Module** to which the relay is connected. **Relay Channel** can return a value between `Relay 1` and `Relay 8`. If **Relay Channel** returns a value of `Invalid`, the VI did not find the relay you specified, and this VI returns an error.

Relay Direction returns the direction in which a motor corresponding to the relay moves.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

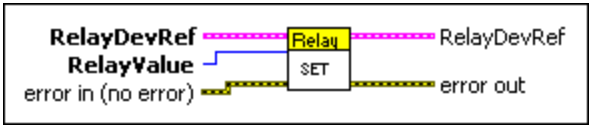
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set.vi

Specifies the direction of the motors corresponding to the relay you specify.



RelayDevRef specifies a reference to the relay you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel specifies the channel of the **DIO Module** to which the relay is connected. **Relay Channel** can specify a value between **Relay 1** and **Relay 8**. If **Relay Channel** is **Invalid**, this VI returns an error.

Relay Direction specifies the direction in which a motor corresponding to the relay moves.

RelayValue specifies whether the relay is configured to move the motor in a forward or reverse direction.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

RelayDevRef returns a reference to the relay.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of **`Slot 4`** or **`Slot 6`**. If **DIO Module** returns a value of **`Default`**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Relay Channel returns the channel of the **DIO Module** to which the relay is connected. **Relay Channel** can return a value between `Relay 1` and `Relay 8`. If **Relay Channel** returns a value of `Invalid`, the VI did not find the relay you specified, and this VI returns an error.

Relay Direction returns the direction in which a motor corresponding to the relay moves.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

33

RobotDrive VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the RobotDrive VIs.

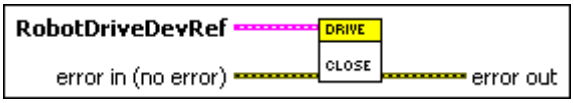
Use the RobotDrive VIs to configure the motors you use to drive the robot. You can use the RobotDrive VIs to drive two-motor or four-motor robots using arcade, tank, or holonomic driving and either joystick or autonomous control. The RobotDrive VIs configure the motors according to the type of motor controller you use.

Use the Open2Motor VI or the Open4Motor VI to specify the PWM channels you want to use for each motor. You then can use the ArcadeDrive VI, HolonomicDrive VI, TankDrive VI, or the RobotDrive Advanced VIs to specify the control mechanism, or the manner in which the PWM values determine the movement of the motors.

Human Interface Device (HID) joysticks map forward movements to negative values. Therefore, if you wire joystick axis values to the RobotDrive VIs, ensure that you specify negative values for forward movements and positive values for backward movements. This issue is a concern mainly when you develop autonomous code. If you wire joystick axis information directly from the Joystick VIs to the RobotDrive VIs, you do not need to manipulate the axis values to move the robot in the intended direction.

Close.vi

Closes the reference to the robot drive you specify. Use this VI to close each robot drive reference that you open with the Open2Motor VI or the Open4Motor VI.



RobotDriveDevRef specifies a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] specifies information about the motor you select.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

Invert specifies, when TRUE, that the motor moves in the reverse direction. When **Invert** is FALSE, the motor moves in the forward direction.

TransformRef specifies a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef specifies a reference to the converter that this VI uses to convert speed values to raw values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot. The range is open-ended, but a range of 0.1 to 1.0 performs well.

Square Inputs specifies, when TRUE, to square the x- and y-values to make the robot easier to maneuver. **Square Inputs** causes the joysticks to have finer control and then to saturate quickly.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in**

(no error) and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

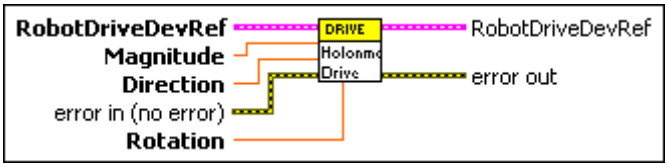
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

HolonomicDrive.vi

Configures the motors for holonomic driving. Holonomic driving allows you to move the robot in any direction without first rotating or maneuvering the robot. This VI requires the robot to have four omni-directional wheels and four motors. Each wheel must rotate at a particular rotational velocity and direction for the robot to obtain and move along a particular heading.

Holonomic drive robots use omni-directional wheels that have a center, large main wheel mounted on a drive shaft and several smaller rollers around the perimeter of the main wheel. The smaller rollers allow the robot to travel in a direction perpendicular to the main wheel without first having to move the main wheel. The robot can travel in any direction by moving the main wheel and the rollers at different speeds.



RobotDriveDevRef specifies a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] specifies information about the motor you select.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

Invert specifies, when TRUE, that the motor moves in the reverse direction. When **Invert** is FALSE, the motor moves in the forward direction.

TransformRef specifies a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef specifies a reference to the converter that this VI uses to convert speed values to raw values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot. The range is open-ended, but a range of 0.1 to 1.0 performs well.

Square Inputs specifies, when TRUE, to square the x- and y-values to make the robot easier to maneuver. **Square Inputs** causes the joysticks to have finer control and then to saturate quickly.

Magnitude specifies the speed at which the robot moves along the **Direction** vector.

Direction specifies the vector, in degrees, along which the robot moves from its original heading.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Rotation specifies the rate, in degrees per second, at which the robot rotates.

RobotDriveDevRef returns a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] returns information about the motor you specify.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot.

Square Inputs indicates, when TRUE, that this parameter squares the x- and y-values.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

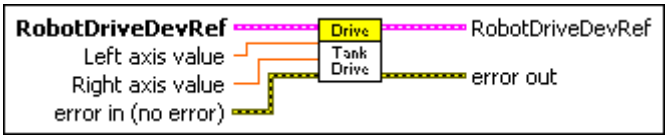
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

TankDrive.vi

Configures the motors for tank driving. Tank driving uses two joysticks where the y-axis values of each joystick specify the power to apply to the motors on either side of the robot.



RobotDriveDevRef specifies a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] specifies information about the motor you select.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

Invert specifies, when TRUE, that the motor moves in the reverse direction. When **Invert** is FALSE, the motor moves in the forward direction.

TransformRef specifies a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef specifies a reference to the converter that this VI uses to convert speed values to raw values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot. The range is open-ended, but a range of 0.1 to 1.0 performs well.

Square Inputs specifies, when TRUE, to square the x- and y-values to make the robot easier to maneuver. **Square Inputs** causes the joysticks to have finer control and then to saturate quickly.

Left Axis Value specifies the power to apply to the left motor.

Right Axis Value specifies the power to apply to the right motor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

RobotDriveDevRef returns a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] returns information about the motor you specify.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot.

Square Inputs indicates, when TRUE, that this parameter squares the x- and y-values.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

34

RobotDrive Advanced VIs

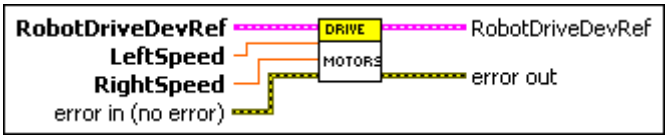
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the RobotDrive Advanced VIs.

Use the RobotDrive Advanced VIs to specify the speed and direction of the motors directly.

Use the Open2Motor VI or the Open4Motor VI to create a **RobotDriveDevRef** reference cluster that you then can wire to the Motors VI on this palette.

Motors.vi

Determines the motor configuration and sends the speed values to the appropriate motors.



RobotDriveDevRef specifies a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] specifies information about the motor you select.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

Invert specifies, when TRUE, that the motor moves in the reverse direction. When **Invert** is FALSE, the motor moves in the forward direction.

TransformRef specifies a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef specifies a reference to the converter that this VI uses to convert speed values to raw values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot. The range is open-ended, but a range of 0.1 to 1.0 performs well.

Square Inputs specifies, when TRUE, to square the x- and y-values to make the robot easier to maneuver. **Square Inputs** causes the joysticks to have finer control and then to saturate quickly.

LeftSpeed specifies the speed for the left motor. Select a value between –1 and 1.

RightSpeed specifies the speed for the right motor. Select a value between –1 and 1.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

RobotDriveDevRef returns a reference to the robot drive you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Motors[] returns information about the motor you specify.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity of the joystick, the more responsive the robot.

Square Inputs indicates, when TRUE, that this parameter squares the x- and y-values.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

35

Sensors VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Sensors VIs.

Use the Sensors VIs to send and receive information from the sensors you connect to the CompactRIO device.

36

Serial Port VIs

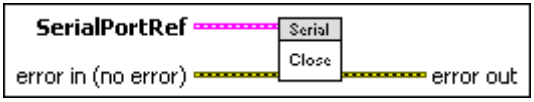
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Serial Port VIs.

Use the Serial Port VIs to read data from or write data to a serial device connected to the serial port of the CompactRIO device.

Use the Open VI to create a **SerialPortRef** reference cluster that you then can wire to the other Serial Port VIs.

Close.vi

Closes the reference to the serial port you specify. Use this VI to close each serial port reference that you open with the Open VI.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

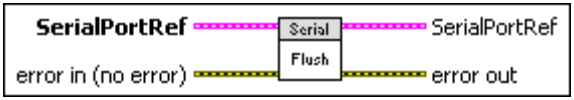
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Flush.vi

Flushes and discards contents of the output buffer of the serial device you specify by writing all the buffered data to the device.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

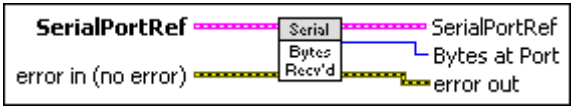
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetBytesReceived.vi

Returns the number of bytes in the input buffer of the serial device you specify.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

Bytes at Port returns the number of bytes in the input buffer of the serial port.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

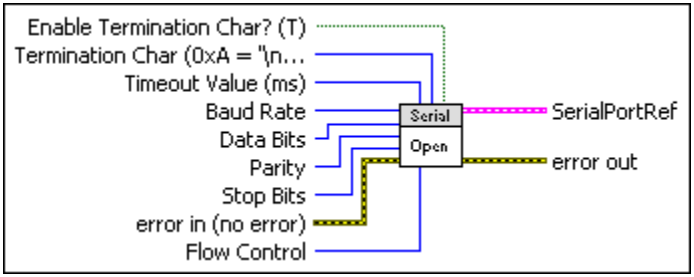
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the serial port. You must open a reference before using any other VIs on this palette.



Enable Termination Char? (T) specifies, when TRUE, that the serial device terminates a read operation when it reads the **Termination Char**. The default is TRUE.

Termination Char (0xA = '\n' = LF) specifies a character that terminates the read operation. The default is **0xA**, which specifies a line feed character.

Timeout Value (ms) specifies the minimum timeout value, in milliseconds, to use for the write and read operations. The default is 5000.

Baud Rate specifies the rate of transmission. The default is 9600.

Data Bits specifies the number of bits in each data character. The value of data bits is between five and eight. The default value is 8.

Parity specifies the parity type to use for the parity bit of each data character you want to transmit or receive.

Stop Bits specifies the number of stop bits that indicate the end of a data character.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Flow Control specifies the type of control used by the transfer mechanism.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Read.vi

Reads a specified number of bytes from the serial device you specify.

SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

Num Bytes to Read specifies the number of bytes to read.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

Buffer returns the data read from the serial device.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

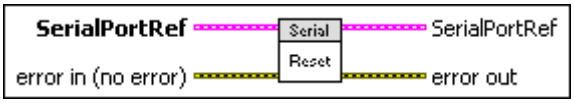
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Reset.vi

Clears the input and output buffers of the serial device you specify.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

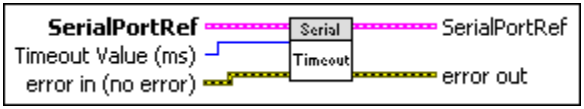
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetTimeout.vi

Specifies the minimum timeout value, in milliseconds, to use when reading data from or writing data to the serial device.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

Timeout Value (ms) specifies the minimum timeout value, in milliseconds, to use for the write and read operations. The default is 5000.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

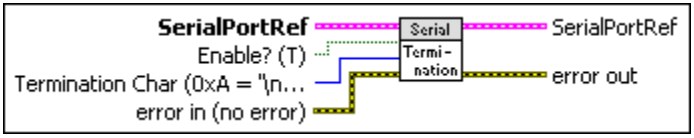
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Termination.vi

Specifies whether to terminate a read operation with a termination character.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

Enable? (T) specifies, when TRUE, that the serial device terminates a read operation when it reads the **Termination Char**. The default is TRUE.

Termination Char (0xA = '\n' = LF) specifies a character that terminates the read operation. The default is **0xA**, which specifies a line feed character.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

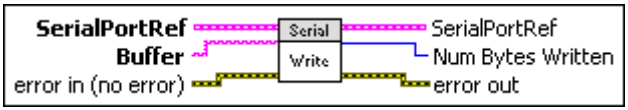
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Write.vi

Writes data to the serial device you specify.



SerialPortRef specifies a reference to the serial port corresponding to the serial device you want to use. Use the VI to open this reference.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Serial Port Name specifies the name of the serial port you want to use.

Buffer specifies the data to write to the serial device.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SerialPortRef returns a reference to the serial port.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Serial Port Name returns the name of the serial port you want to use.

Num Bytes Written returns the number of bytes written to the serial device.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

37

Servo VIs

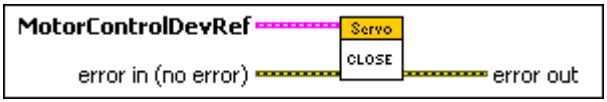
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Servo VIs.

Use the Servo VIs to set or get the angle and position of a servo. If you use a servo, use the Servo VIs instead of the PWM VIs. The Servo VIs use preconfigured PWM range and deadband values specific to the servo so you do not need to perform any conversions or scaling calculations on the raw PWM values.

Use the Open VI to create a **PWMDeviceRef** reference cluster that you then can wire to the other Servo VIs.

Close.vi

Closes the reference to the servo controller you specify. Use this VI to close each servo reference that you open with the Open VI.



MotorControlDevRef specifies a reference to the motor controller you want to use.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

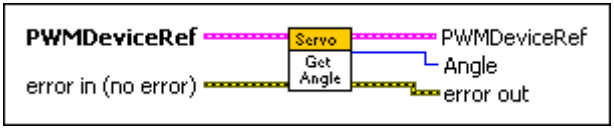
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetAngle.vi

Returns the angle of the servo you specify.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other VIs, use the VI on the **Servo** palette to open this reference. Otherwise, use the general PWM VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Angle returns the angle, in degrees, of the servo.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

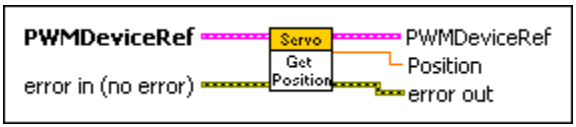
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetPosition.vi

Returns the position of the servo you specify.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other VIs, use the VI on the **Servo** palette to open this reference. Otherwise, use the general PWM VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

Position returns the position of the servo. **Position** ranges from 0.0 to 1.0.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

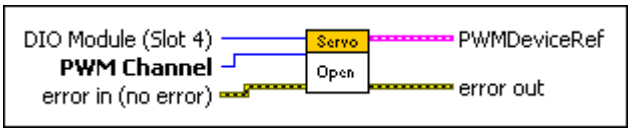
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Creates a reference to the servo controller you specify. You must open a reference before using any other VIs on this palette.



DIO Module (Slot 4) specifies the slot number on the CompactRIO device of the digital module you want to use. Select **Slot 4** or **Slot 6**. The default is **Slot 4**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. Select a value between **PWM 1** and **PWM 10**. The default is **PWM 1**. If **PWM Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is

normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

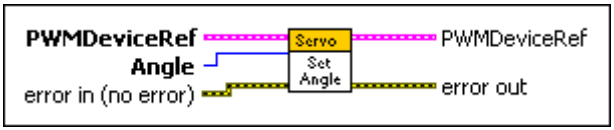
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetAngle.vi

Sets the angle of the servo you specify.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other VIs, use the VI on the **Servo** palette to open this reference. Otherwise, use the general PWM VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Angle specifies the angle, in degrees, of the servo.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

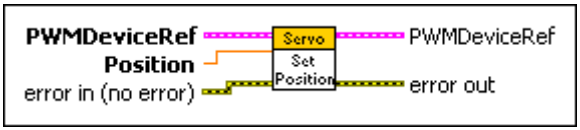
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetPosition.vi

Specifies the position of the servo you specify.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other VIs, use the VI on the **Servo** palette to open this reference. Otherwise, use the general PWM VI to open this reference.

DeviceStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel specifies the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can specify a value between **PWM 1** and **PWM 10**. If **PWM Channel** is **Invalid**, this VI returns an error.

TransformRef specifies a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef specifies a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand specifies the range of the deadband for the signal. The deadband is a range in the signal that has no effect on the motor.

Name identifies the PWM signal and deadband.

maxPositivePwm specifies the maximum value of the signal.

minPositivePwm specifies the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm specifies the center value of the PWM signal.

maxNegativePwm specifies the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm specifies the minimum value of the signal.

angularRange specifies the maximum range of the servo, if applicable.

Position specifies the position of the servo. **Position** ranges from 0.0 to 1.0.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

PWMDeviceRef returns a reference to the PWM. The PWM specifies the amount of power sent to the motors.

DeviceStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can return a value of `Slot 4` or `Slot 6`. If **DIO Module** returns a value of `Default`, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

PWM Channel returns the channel of the **DIO Module** from which you want to read the PWM value. **PWM Channel** can return a value between `PWM 1` and `PWM 10`. If **PWM Channel** returns a value of `Invalid`, the VI did not find the digital channel you specified, and this VI returns an error.

Invert returns whether the motor moves in the forward (FALSE) or reverse (TRUE) direction.

TransformRef returns a reference to the converter that the PWM uses to convert PWM values to scaled PWM values.

InvTransformRef returns a reference to the converter that the PWM uses to convert scaled PWM values to PWM values.

DeadBand returns the range of the deadband for the signal.

Name returns a reference for the PWM signal.

maxPositivePwm returns the maximum value of the signal.

minPositivePwm returns the minimum positive value of the signal that creates a response. **minPositivePwm** is the maximum value of the deadband.

centerPwm returns the center value of the PWM signal.

maxNegativePwm returns the maximum negative value of the signal that creates a response. **maxNegativePwm** is the minimum value of the deadband.

minNegativePwm returns the minimum value of the signal.

angularRange returns the maximum range of the servo, if applicable.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

38

Solenoid VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Solenoid VIs.

Use the Solenoid VIs to operate a solenoid. A solenoid is a pneumatic valve that can release air to move a part on the robot.

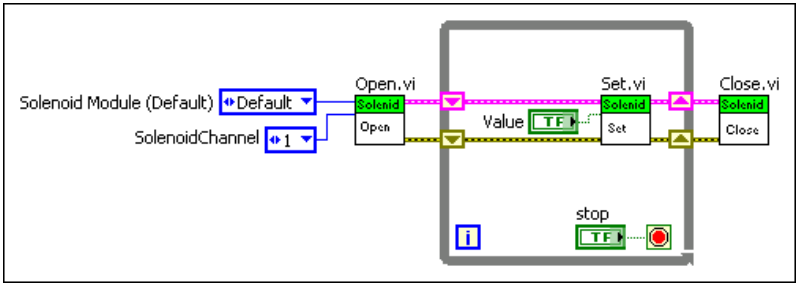
Use the Open VI to create a **SolenoidDevRef** reference cluster that you then can wire to the other Solenoid VIs.

You connect the solenoid to the NI 9472 digital output module on the CompactRIO device. You then use the Solenoid VIs to specify whether the NI 9472 sends a 12 V signal to the solenoid. When the solenoid receives the 12 V signal, the solenoid releases air, thus moving the robot part.

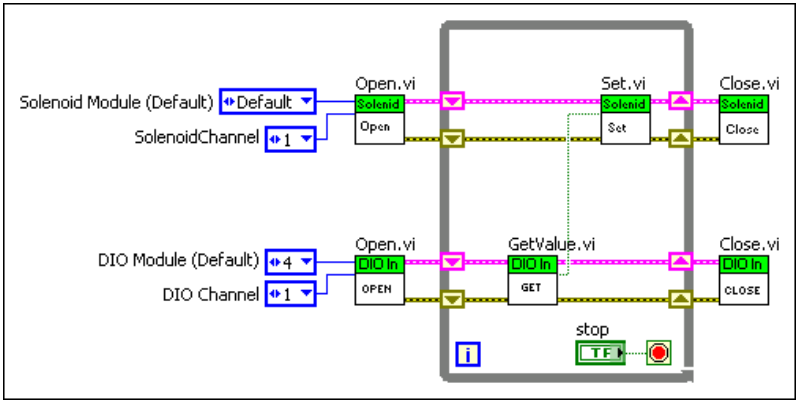
Each time the solenoid releases air, the air pressure of the compression system drops. If the air pressure drops below a certain level, you can use the Compressor VIs to turn on the pressure switch of the compressor. The compressor then pumps more air into the system, thus increasing the air pressure.

If you want to operate a solenoid but all channels of the NI 9472 are used, you alternatively can connect a relay to the NI 9403 digital input/output module and then connect the relay to the solenoid. The NI 9403 cannot output sufficient voltage to operate the solenoid. However, the NI 9403 can operate the relay. The relay connects the solenoid to the battery, which can provide a greater voltage current to operate the solenoid.

The following figure demonstrates how to control a solenoid connected to solenoid channel 1 of the default solenoid module on the CompactRIO device. You can use this example VI to operate a solenoid continuously.

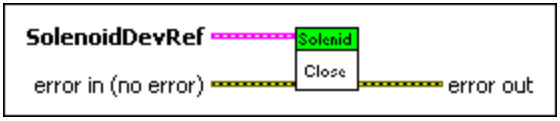


The following figure demonstrates how to set the value of a solenoid based on a digital input value. A solenoid is connected to channel 1 of the default solenoid module on the CompactRIO device. A digital line is connected to channel 1 of the digital module in slot 4 of the CompactRIO device. The GetValue VI reads a digital input value and passes it to the Set VI, which in turn specifies whether the NI 9472 sends a 12 V signal to the solenoid.



Close.vi

Closes the reference to the solenoid you specify. Use this VI to close each solenoid reference that you open with the Open VI.



SolenoidDevRef specifies a reference to the solenoid you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Solenoid Module specifies the slot number on the CompactRIO device of the solenoid module you want to use. **Solenoid Module** always specifies a value of **Slot 8**.

Solenoid Channel specifies the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can specify a value between **Solenoid 1** and **Solenoid 8**. If **Solenoid Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

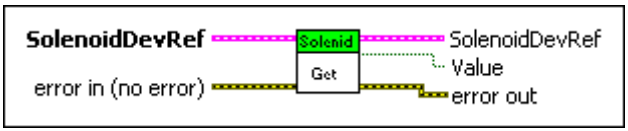
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Get.vi

Returns whether you specified the NI 9472 to send a 12 V signal to the solenoid you specify. When the solenoid receives the 12 V signal, the solenoid releases air, thus moving the associated robot part.



SolenoidDevRef specifies a reference to the solenoid you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Solenoid Module specifies the slot number on the CompactRIO device of the solenoid module you want to use. **Solenoid Module** always specifies a value of **Slot 8**.

Solenoid Channel specifies the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can specify a value between **Solenoid 1** and **Solenoid 8**. If **Solenoid Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SolenoidDevRef returns a reference to the solenoid.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Solenoid Module returns the slot number on the CompactRIO device of the solenoid module to which the solenoid is connected. **Solenoid Module** always returns `Slot 8`.

Solenoid Channel returns the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can return a value between `Solenoid 1` and `Solenoid 8`. If **Solenoid Channel** returns a value of `Invalid`, the VI did not find the solenoid you specified, and this VI returns an error.

Value returns TRUE if you specified the NI 9472 to send a 12 V signal to the solenoid you specify. Otherwise, **Value** returns FALSE.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

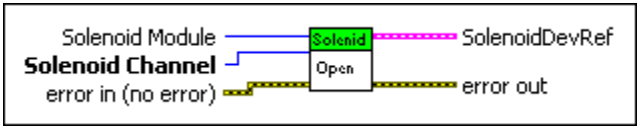
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the solenoid you specify. You must open a reference before using any other VIs on this palette.



Solenoid Module specifies the slot number on the CompactRIO device of the solenoid module you want to use. You can select only **Slot 8**.

Solenoid Channel specifies the channel of the **Solenoid Module** to which the solenoid is connected. You can select a value between **Solenoid 1** and **Solenoid 8**. If **Solenoid Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SolenoidDevRef returns a reference to the solenoid.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Solenoid Module returns the slot number on the CompactRIO device of the solenoid module to which the solenoid is connected. **Solenoid Module** always returns `Slot 8`.

Solenoid Channel returns the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can return a value between `Solenoid 1` and `Solenoid 8`. If **Solenoid Channel** returns a value of `Invalid`, the VI did not find the solenoid you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

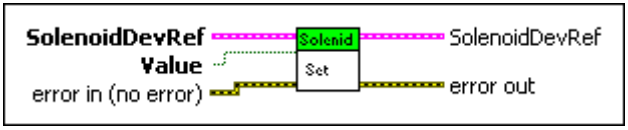
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Set.vi

Specifies whether the NI 9472 sends a 12 V signal to the solenoid you specify. When the solenoid receives the 12 V signal, the solenoid releases air, thus moving the associated robot part.



SolenoidDevRef specifies a reference to the solenoid you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Solenoid Module specifies the slot number on the CompactRIO device of the solenoid module you want to use. **Solenoid Module** always specifies a value of **Slot 8**.

Solenoid Channel specifies the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can specify a value between **Solenoid 1** and **Solenoid 8**. If **Solenoid Channel** is **Invalid**, this VI returns an error.

Value specifies, when TRUE, that the NI 9472 sends a 12 V signal to the solenoid.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SolenoidDevRef returns a reference to the solenoid.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Solenoid Module returns the slot number on the CompactRIO device of the solenoid module to which the solenoid is connected. **Solenoid Module** always returns **`Slot 8`**.

Solenoid Channel returns the channel of the **Solenoid Module** to which the solenoid is connected. **Solenoid Channel** can return a value between **`Solenoid 1`** and **`Solenoid 8`**. If **Solenoid Channel** returns a value of **`Invalid`**, the VI did not find the solenoid you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

39

SPI VIs

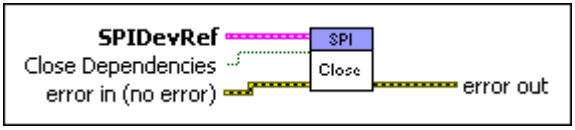
Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the SPI VIs.

Use the Serial Peripheral Interface (SPI) VIs to communicate with sensors or custom circuits. The hardware currently supports only Master Mode SPI.

Use the Open VI to create an **SPIDevRef** reference cluster that you then can wire to the other SPI VIs.

Close.vi

Closes a reference to the SPI Engine. Use this VI to close each SPI reference that you open with the Open VI.



SPIDevRef specifies a reference to the SPI Engine you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Close Dependencies specifies to close any digital inputs when you close the reference to the SPI Engine. When TRUE, **Close Dependencies** closes the digital inputs.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

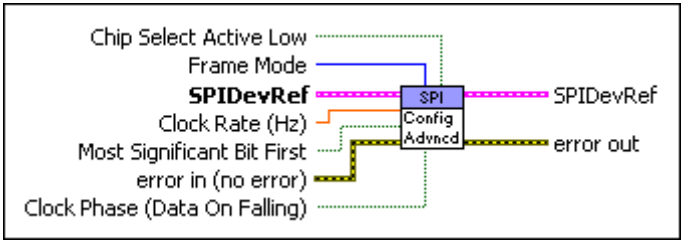
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

ConfigureAdvancedOptions.vi

Allows you to set more advanced SPI controls. Depending on the definition of your interface, this VI might not be required.



Chip Select Active Low specifies, when TRUE, that the active state of the chip select output line is low. When FALSE, **Chip Select Active Low** specifies that the active state of the chip select output line is high.

Frame Mode specifies the behavior of the chip select line in relation to the duration of the frame.

SPIDevRef specifies a reference to the SPI Engine you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Clock Rate (Hz) specifies the rate of the generated clock signal. The default and maximum value is 76,628 Hz.

Most Significant Bit First specifies the the order the SPI VIs send and receive the bits. When TRUE, the most significant bit is sent first. When FALSE, the most significant bit is sent last.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Clock Phase (Data On Falling) specifies, when TRUE, that the data is stable on the falling edge and the data changes on the rising edge. When FALSE, **Clock Phase (Data On Falling)** specifies that the data changes on the falling edge and the data is stable on the rising edge.

SPIDevRef returns a reference to the SPI Engine.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select

Explain Error from the shortcut menu for more information about the error.

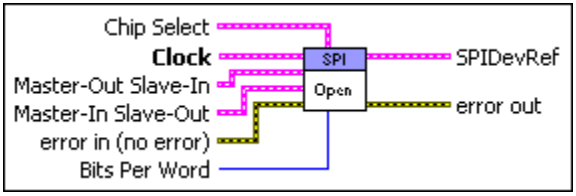
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the SPI Engine. You must open a reference before using any other VIs on this palette. After you open a reference to the SPI Engine, you can use the other SPI VIs to access and monitor the sensors or custom circuits from the SPI Engine.



Chip Select specifies a digital output reference that outputs according to **Frame Mode** and **ChipSelectActiveLow** from the VI.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

Clock specifies a digital output reference that outputs the clock signal.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

Master-Out Slave-In specifies a digital output reference that outputs the written data to the slave.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

Master-In Slave-Out specifies a digital input reference that inputs the data from the slave.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

DIO Module specifies the slot number on the CompactRIO device of the digital module you want to use. **DIO Module** can specify a value of **Slot 4** or **Slot 6**. If **DIO Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

DIO Channel specifies the channel of the **DIO Module** that you want to use. **DIO Channel** can specify a value between **DIO 1** and **DIO 14**. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Bits Per Word specifies the number of bits from each word that the slave transmits or receives. **Bits Per Word** defines the number of bits in one frame. The range of **Bits Per Word** is one to 32 bits.

SPIDevRef returns a reference to the SPI Engine.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

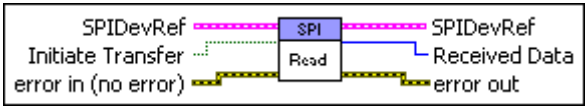
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Read.vi

Reads a word from the slave device.



SPIDevRef specifies a reference to the SPI Engine you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Initiate Transfer specifies, when TRUE, that this VI pushes a word into the transmit buffer to initiate a transfer. When FALSE, **Initiate Transfer** specifies that data is already in the buffer from a previous write.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SPIDevRef returns a reference to the SPI Engine.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Received Data returns one word from the slave device.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

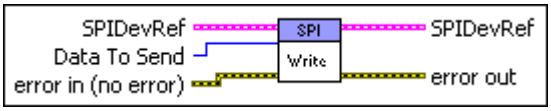
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Write.vi

Writes a word to the slave device.



SPIDevRef specifies a reference to the SPI Engine you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Data To Send specifies one word to write to the slave device.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in**

(no error) and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

SPIDevRef returns a reference to the SPI Engine.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

40

Ultrasonic VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Ultrasonic VIs.

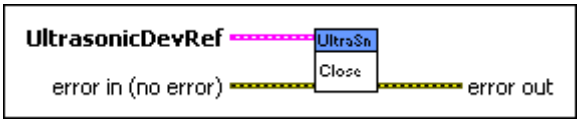
Use the Ultrasonic VIs to measure the distance between an ultrasonic sensor on the robot and the closest object in front of the sensor.

An ultrasonic sensor emits a ping that bounces off the closest object and returns to the sensor as an echo. The Ultrasonic VIs support only ultrasonic sensors that contain both a ping channel and an echo channel where the signal on the echo channel remains high while the sensor waits for an echo.

The signal on the echo channel is low by default. When you use the Ping VI to emit a ping from an ultrasonic sensor, the signal on the echo channel goes high. The ping bounces off the closest object in front of the sensor, and the sensor receives the echo of the ping. The echo channel then returns low. A counter on the CompactRIO device determines the time the signal on the echo channel is high. You then can use the GetRange VI to determine the distance, in inches or millimeters, between the sensor and the object.

Close.vi

Closes the reference to the ultrasonic sensor you specify. Use this VI to close each ultrasonic sensor reference that you open with the Open VI.



UltrasonicDevRef specifies a reference to the ultrasonic sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Ping Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can specify a value of **4** or **6**. If **Ping Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel specifies the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can specify a value between 1 and 14. If **Ping Channel** is **Invalid**, this VI returns an error.

Echo Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can specify a value of **4** or **6**. If **Echo Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel specifies the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can specify a value between 1 and 14. If **Echo Channel** is **Invalid**, this VI returns an error.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

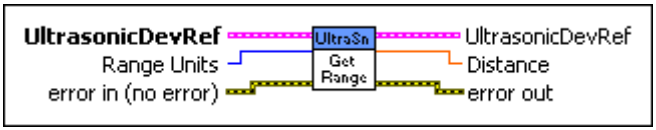
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetRange.vi

Returns the distance, in inches or millimeters, between the ultrasonic sensor and the closest object in front of the sensor. The ultrasonic sensor measures this distance by emitting a ping and calculating the time it takes for the echo of the ping to return.



UltrasonicDevRef specifies a reference to the ultrasonic sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Ping Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can specify a value of **4** or **6**. If **Ping Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel specifies the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can specify a value between 1 and 14. If **Ping Channel** is **Invalid**, this VI returns an error.

Echo Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can specify a value of **4** or **6**. If **Echo Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel specifies the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can specify a value between 1 and 14. If **Echo Channel** is **Invalid**, this VI returns an error.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

Range Units specifies the units in which you want to measure the distance between the ultrasonic sensor and the closest object in front of the sensor.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

UltrasonicDevRef returns a reference to the ultrasonic sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Ping Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can return a value of **`Slot 4`** or **`Slot 6`**. If **Ping Module** returns a value of **`Default`**, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel returns the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can return a value between 1 and 14. If **Ping Channel** returns a value of **`Invalid`**, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

Echo Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can return a value of **`4`** or **`6`**. If **Echo Module** returns a value of **`Default`**, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel returns the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can return a value between 1 and 14. If **Echo Channel** returns a value of **`Invalid`**, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between **`Ctr 0`** and **`Ctr 7`**. If **CntIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

Distance returns the distance, in the **Range Units** you specified, between the ultrasonic sensor and the closest object in front of the sensor.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

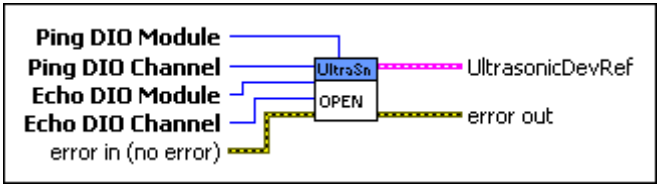
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Creates a reference to the ultrasonic sensor you specify. You must open a reference before using any other VIs on this palette.



Ping DIO Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. Select **4** or **6**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo DIO Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. Select **4** or **6**. You also can select **Default** to specify the slot of the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping DIO Channel specifies the channel on the **Ping DIO Module** to which the ultrasonic sensor that emits the ping is connected. Select a value between 1 and 14. If **Ping DIO Channel** is **Invalid**, this VI returns an error.

Echo DIO Channel specifies the channel on the **Echo DIO Module** to which the ultrasonic sensor that receives the echo of the ping is connected. Select a value between 1 and 14. If **Echo DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

UltrasonicDevRef returns a reference to the ultrasonic sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Ping Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can return a value of `Slot 4` or `Slot 6`. If **Ping Module** returns a value of `Default`, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel returns the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can return a value between 1 and 14. If **Ping Channel** returns a value of `Invalid`, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

Echo Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can return a value of `4` or `6`. If **Echo Module** returns a value of `Default`, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel returns the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can return a value between 1 and 14. If **Echo Channel** returns a value of `Invalid`, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between `Ctr 0` and `Ctr 7`. If **CntIndex** returns a value of `Invalid`, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

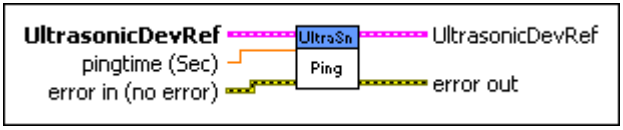
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Ping.vi

Emits a single ping from the ultrasonic sensor you specify.



UltrasonicDevRef specifies a reference to the ultrasonic sensor you want to use.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Ping Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can specify a value of **4** or **6**. If **Ping Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel specifies the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can specify a value between 1 and 14. If **Ping Channel** is **Invalid**, this VI returns an error.

Echo Module specifies the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can specify a value of **4** or **6**. If **Echo Module** specifies a value of **Default**, this VI uses the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel specifies the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can specify a value between 1 and 14. If **Echo Channel** is **Invalid**, this VI returns an error.

CntIndex specifies the index of the reserved counter. **CntIndex** can specify a value between **Ctr 0** and **Ctr 7**. If **CntIndex** is **Invalid**, this VI returns an error.

pingtime (Sec) specifies the duration, in seconds, of the ping the ultrasonic sensor emits. The default is 1E-5.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is

normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

UltrasonicDevRef returns a reference to the ultrasonic sensor.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Ping Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that emits the ping is connected. **Ping Module** can return a value of `Slot 4` or `Slot 6`. If **Ping Module** returns a value of `Default`, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Ping Channel returns the channel on the **Ping Module** to which the ultrasonic sensor that emits the ping is connected. **Ping Channel** can return a value between 1 and 14. If **Ping Channel** returns a value of `Invalid`, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

Echo Module returns the digital module on the CompactRIO device to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Module** can return a value of **`4`** or **`6`**. If **Echo Module** returns a value of **`Default`**, the ultrasonic sensor is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4.

Echo Channel returns the channel on the **Echo Module** to which the ultrasonic sensor that receives the echo of the ping is connected. **Echo Channel** can return a value between 1 and 14. If **Echo Channel** returns a value of **`Invalid`**, the VI did not find the ultrasonic sensor you specified, and this VI returns an error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between **`Ctr 0`** and **`Ctr 7`**. If **CntIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

41

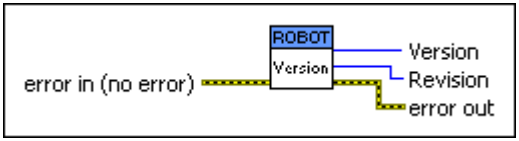
Utilities VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Utilities VIs.

Use the Utilities VIs to return information about the CompactRIO device, such as the temperature of the chassis, the state of the LEDs, and the version of the FPGA.

FRC FPGAVersion.vi

Returns the version and revision of the FPGA image on the CompactRIO device.



error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Version returns the version of the FPGA image. For the *FIRST* Robotics Competition (FRC), the version of the FPGA image corresponds to the year of the competition.

Revision returns the revision of the FPGA image in the format **`0xAAABBCCC`**, where *A*, *B*, and *C* correspond to the revision `A.B.C`. For example, **Revision** returns **`0x0010A00F`** for revision 1.10.15.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

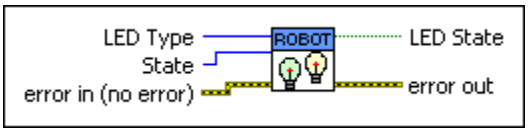
code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FRC LEDs.vi

Sets the state, either on or off, of the USER1 LED or the FPGA LED on the CompactRIO device. You can define both the USER1 LED and the FPGA LED to meet the needs of your application.

Note If you set the **State** of the USER1 LED or the FPGA LED to **Toggle**, the LED changes to the opposite of the current state. If you want the LED to blink, you must call this VI continuously.



LED Type specifies the LED whose state you want to set.

State specifies the state, either on or off, to which you want to set the LED you specify with the **LED Type** control.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

LED State returns TRUE if the LED is on and FALSE if the LED is off.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

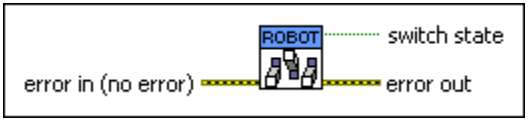
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

FRC ReadSwitch.vi

Returns the position of the USER1 switch on the CompactRIO device. Use the RT Read Switch VI to define the USER1 switch for your application.



error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

switch state returns TRUE if the USER1 switch is in the ON position and FALSE if the USER1 switch is in the OFF position.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

42

Watchdog VIs

Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for the latest information about the Watchdog VIs.

Use the Watchdog VIs to control the user watchdog if you choose to enable the user watchdog.

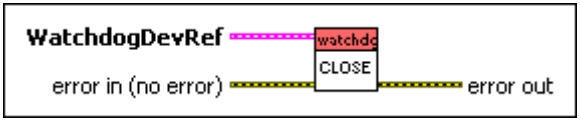
Each CompactRIO device consists of two watchdogs, a system watchdog and a user watchdog. The watchdogs ensure that the robot does not continue moving if the robot loses communication with the driver station or field management system (FMS). Each watchdog can disable the PWM, relay, and solenoid outputs of the CompactRIO device if the watchdog times out. A watchdog times out if it is not fed during a specified amount of time.

The system watchdog is always enabled. During the *FIRST* Robotics Competition (FRC), the system watchdog is alive as long as it receives an "enabled" signal from the driver station. The system watchdog can time out if the driver station sends a "disabled" signal, if a network failure occurs, or if the driver station is in an emergency stop, or e-stop, state. You can use the SetEnabled VI to specify whether the user watchdog is enabled. If you enable the user watchdog, you must specify a timeout period and ensure that the program you write feeds the watchdog regularly.

Use the Open VI to create a **WatchdogDevRef** reference cluster that you then can wire to the other Watchdog VIs.

Close.vi

Closes the reference to the user watchdog you specify. Use this VI to close each watchdog reference that you open with the Open VI.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

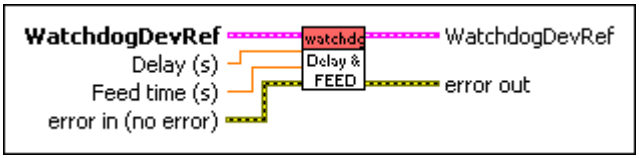
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Delay and Feed.vi

Delays execution for a specified amount of time but continues feeding the user watchdog to avoid timing out. You can use this VI in autonomous code to hold the settings of the motors for a certain amount of time without the user watchdog timing out.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Delay (s) specifies the amount of time, in seconds, for which to delay execution. The default is 0.

Feed time (s) specifies the interval, in seconds, at which to feed the user watchdog while execution is delayed. The default is 0.05.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

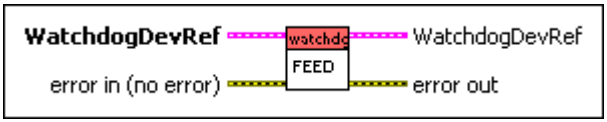
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Feed.vi

Sends a signal to, or feeds, the user watchdog.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

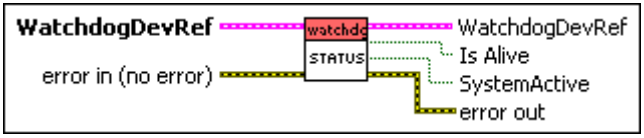
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

GetStatus.vi

Returns whether the user watchdog is alive and whether the system is active. The system is active when both the system and user watchdogs are either disabled or enabled and alive.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Is Alive returns TRUE if the user watchdog is alive. Otherwise, **Is Alive** returns FALSE.

SystemActive returns TRUE if the system is active. The system is active when both the system and user watchdogs are either disabled or enabled and alive. If either watchdog is enabled and has timed out, **SystemActive** returns FALSE.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

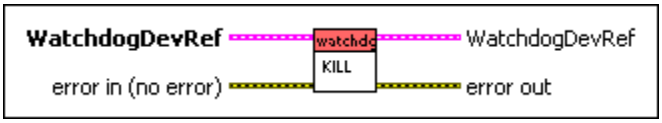
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Kill.vi

Forces the user watchdog to time out immediately. When the user watchdog times out, it disables the PWM, relay, and solenoid outputs of the CompactRIO device.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

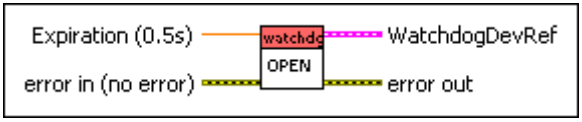
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

Open.vi

Opens a reference to the user watchdog. You must open a reference before using any other VIs on this palette.



Expiration (0.5s) specifies the time, in seconds, that the user watchdog waits to be fed. If the user watchdog is not fed during this time, it disables the PWM, relay, and solenoid outputs of the CompactRIO device. The default is 0.5 seconds.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

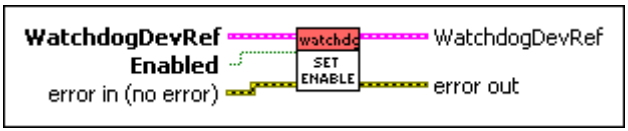
status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

SetEnabled.vi

Specifies whether the user watchdog is enabled. If the user watchdog is disabled, the system is still active as long as the system watchdog also is disabled or is enabled and alive. Furthermore, if the user watchdog is disabled, using the Kill VI does not disable the PWM, relay, or solenoid outputs of the CompactRIO device.



WatchdogDevRef specifies a reference to the user watchdog.

DevStatus describes the error status before this VI or function runs. The default is no error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

Enabled specifies, when TRUE, that the user watchdog is enabled. The default is TRUE. If the user watchdog is enabled, the application you develop must account for feeding the watchdog at regular periods. If the user watchdog is not fed during the expiration period you specify with the VI, the watchdog disables the PWM, relay, and solenoid outputs of the CompactRIO device.

error in (no error) describes error conditions that occur before this node runs. The default is `no error`. If an error occurred before this node runs, the node passes the **error in (no error)** value to **error out**. This node runs normally only if no error occurred before this node runs. If an error occurs while this node runs, it runs normally and sets its own error status in **error out**. Use the Simple Error Handler or General Error Handler VIs to display the description of the error code. Use exception control to treat what is normally an error as no error or to treat a warning as an error. Use **error in (no error)** and **error out** to check errors and to specify execution order by wiring **error out** from one node to **error in (no error)** of the next node.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning. The default is an empty string.

WatchdogDevRef returns a reference to the user watchdog.

DevStatus describes the error status that this VI or function produces.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.

error out contains error information. If **error in** indicates that an error occurred before this VI or function ran, **error out** contains the same error information. Otherwise, it describes the error status that this VI or function produces. Right-click the **error out** front panel indicator and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred before this node ran or FALSE (checkmark) to indicate a warning or that no error occurred before this node ran.

code is the error or warning code. If **status** is TRUE, **code** is an error code. If **status** is FALSE, **code** is 0 or a warning code.

source specifies the origin of the error or warning and is, in most cases, the name of the node that produced the error or warning.