LabVIEW™

Robotics Programming Guide for the *FIRST* Robotics Competition

October 2008
372668A-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.

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

Trademarks

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

Chapter 3
Configuring the Camera and the CompactRIO Device

Chapter 4
Using the FRC Framework

Chapter 5
Tutorial: Creating an FRC cRIO Robot Project

Chapter 6
Using the WPI Robotics Library VIs

Chapter 7
Accelerometer VIs

Chapter 8
Accumulator VIs

Chapter 9
Actuators VIs

Chapter 10
AnalogChannel VIs

Chapter 11
AnalogChannel Advanced VIs

Chapter 12
Analog Trigger VIs

Chapter 13
Camera VIs

Chapter 14
Camera Properties VIs

Chapter 15
Compressor VIs

Chapter 16
Counter VIs

Chapter 17
DigitalInput VIs

Chapter 18
DigitalOutput VIs

Chapter 19
Digital Output Advanced VIs

Chapter 20
DriverStation VIs

Chapter 21
Encoder VIs

Chapter 22
Gyro VIs

Chapter 23
IO VIs

Chapter 24
Jaguar VIs

Chapter 25
Joystick VIs

Chapter 26
MotorControl VIs

Chapter 27
PWM VIs

Chapter 28
Relay VIs

Chapter 29
RobotDrive VIs

Chapter 30
Sensors VIs

Chapter 31
Servo VIs

Chapter 32
Solenoid VIs

Chapter 33
Ultrasonic VIs

Chapter 34
Utilities VIs

Chapter 35
Victor VIs

Chapter 36
Watchdog VIs

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. The FRC cRIO robot project contains Case structures to handle each status. Refer to the *FRC cRIO Robot Project* section of Chapter 4, *Using the FRC Framework*, for more information about the FRC cRIO 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 bumpers, 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 cRIO 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

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

Slot	I/O Module
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 bumper to each of the NI 9201 modules. You then can connect any analog sensors, such as accelerometers and gyros, to the analog bumpers to acquire analog data.

Use the Accelerometer VIs and the Gyro VIs to acquire analog data from those sensors. Refer to Chapter 7, *Accelerometer VIs*, and Chapter 22, *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 10, *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 16, *Counter VIs*, Chapter 21, *Encoder VIs*, and Chapter 33, *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 17, *DigitalInput VIs*, and Chapter 18, *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 bumper 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 bumper to the NI 9472 and connect the solenoid to the pneumatic bumper. Then use the Solenoid VIs to specify whether the NI 9472 sends a 12 V signal to the solenoid.

Refer to Chapter 32, *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 13, *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 13, *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 7 through 36 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.

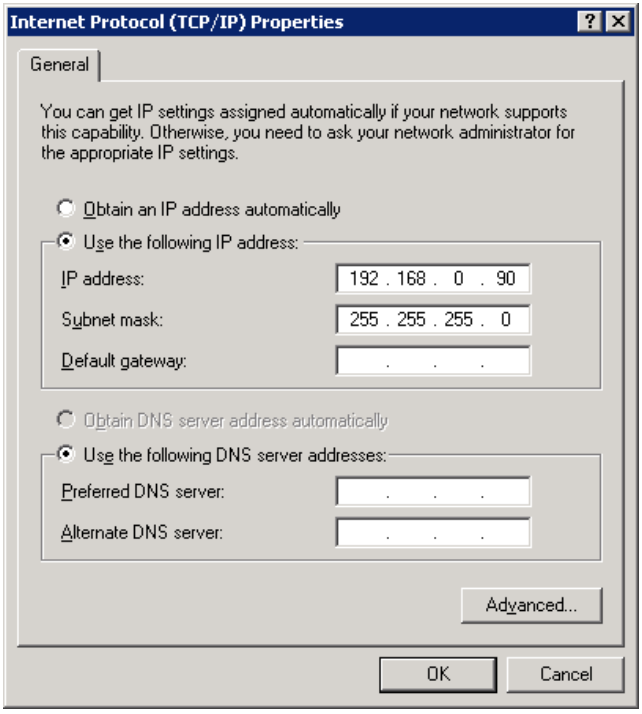
To configure the Axis camera, you must set the static IP address of the computer, connect the computer to the camera, and then run the Setup Axis Camera Tool.

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 `192.168.0.90`.

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.

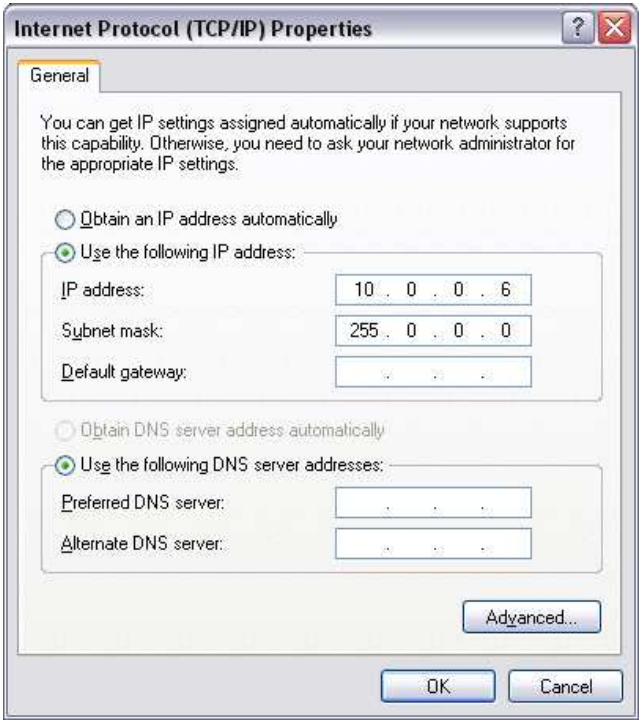
1. Select **Start»Control Panel»Network Connections»Local Area Connection** to display the **Local Area Connection Status** dialog box.
2. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
3. On the **General** page, select **Internet Protocol (TCP/IP)** 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.x.y.6`, where *y* corresponds to the last two digits of the team number and *x* 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.

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 modify the LabVIEW or C/C++ program 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, team ID, or robot number.
5. From the **Select Image** list, select `FRC_2009_v1.zip` to download the FRC_2009_v1 image to the CompactRIO device. The FRC_2009_v1 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.`*`x`*`.`*`y`*`.2`, where *y* corresponds to the last two digits of the team number and *x* corresponds to the first or first two digits of the team number.
8. Select **1** as the number to assign to the robot in the **Robot Number** field.

9. Click the **Apply** button to apply the changes you made and download the FRC_2009_v1 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.
10. 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_v1 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.

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 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 cRIO 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 cRIO Robot Project

The FRC cRIO robot project template starts communication between the CompactRIO device and the driver station, initializes the user watchdog and the camera, and provides Case structures that you can use to handle each competition mode and status of the robot.

Use the **Create New FRC cRIO Robot Project** dialog box, available by clicking the **FRC cRIO Robot Project** link in the **Getting Started** window, to create an FRC cRIO robot project. Refer to Chapter 5, *Tutorial: Creating an FRC cRIO Robot Project*, for more information about creating an FRC cRIO robot project.

The FRC cRIO robot project looks similar to the following figure.

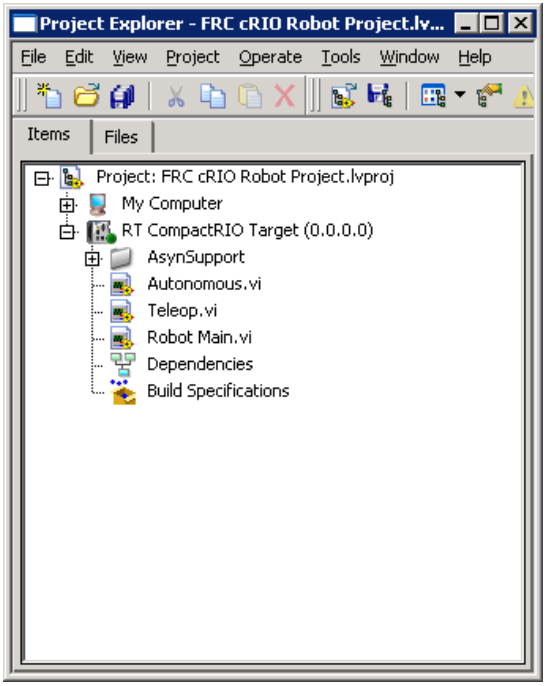


Figure 4-1. New FRC cRIO Robot Project

The FRC cRIO 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. This target contains three top-level VIs: Autonomous VI, Teleop VI, and Robot Main VI.

Robot Main VI

The Robot Main VI is the master VI for the robot. This VI establishes communication with the driver station, initializes the user watchdog to an enabled state, starts acquiring image data with the Axis camera, and calls the Teleop VI or the Autonomous VI 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 two While Loops.

The first While Loop determines the behavior of the robot, such as how the motors respond to sensor data. The Start Communication VI and the Watchdog Open VI run before this 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. The user watchdog ensures that the robot does not continue moving if the program stops executing. At each iteration of the first While Loop, the Feed VI feeds the user watchdog.

Notice that the first While Loop contains a Case structure with a case for each competition state (Autonomous Disabled, Autonomous Enabled, TeleOp Disabled, TeleOp Enabled). Each competition state contains another Case structure to handle the three possible derived states of the robot: Init, Execute, or Stop.

Refer to Chapter 1, *Overview of the FIRST Robotics Competition*, for more information about the competition states and derived states.

The Robot Main VI contains default block diagram code for all states in the Enabled status. For example, in the Autonomous Enabled case, if the robot state is Execute, the Robot Main VI passes an operation of **Execute** to the Autonomous VI. Thus, the autonomous program you want to run executes. If the robot state is Stop, the Robot Main VI passes an operation of **Stop** to the Autonomous VI, and the autonomous program stops. You can add *FIRST* Robotics Competition VIs and other LabVIEW VIs to the Robot Main VI to handle each of the competition states and derived states.

When the first While Loop stops, the Kill VI kills the user watchdog. 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.

The second While Loop processes image data that you acquire from an Axis camera. Before the second While Loop runs, the Camera Open and Start VIs open a reference to the Axis camera and start acquiring image data from the camera, respectively. If you choose to enable image processing, the second While Loop uses the Get Image VI to retrieve a specific image from the image data that the camera acquired. You then can use the *FIRST* Vision VIs to process the 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.

Teleop VI

The Teleop VI contains the program you want to use during the TeleOp mode of the competition. In TeleOp mode, the robot moves in response to commands it receives from input devices, such as joysticks and game controllers. You can use the Teleop VI to program the robot to move motors or sensors according to input device data and to pass this data to the host computer through the driver station.

In the **Project Explorer** window, double-click the **Teleop.vi** item to open the Teleop VI. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. Notice that the Teleop VI consists of a While Loop and a Case structure.

By default, the Init case of the Teleop VI reads data from the joystick connected to port 1 of the driver station and opens a reference to the motors. This VI uses the Open2WheelWithVictor VI, which assumes the robot uses Victor motor controllers to control the motors. The Execute case of the Teleop VI moves the robot according to axis information from the joystick. The Close case closes the reference to the motors. You can add *FIRST* Robotics Competition VIs and other LabVIEW VIs to the Teleop VI to specify which input devices you want to use and how you want the robot to behave according to the data those devices provide.

Autonomous VI

The Autonomous VI contains the program you want to use during the Autonomous mode of the competition. In Autonomous mode, the robot moves without receiving signals from input devices.

In the **Project Explorer** window, double-click the **Autonomous.vi** item to open the Autonomous VI. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. Notice that the Autonomous VI consists of a While Loop and a Case structure.

The Case structure contains three cases: Init, Execute, and Stop. These cases correspond to the robot states. Add *FIRST* Robotics Competition VIs and other LabVIEW VIs to each case to specify how you want the robot to behave in the corresponding robot state.

Deploying the FRC cRIO Robot Project

After you develop the FRC cRIO 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 VI you want to deploy to the CompactRIO device. LabVIEW deploys the VI, all items required by the VI, and the target settings to memory on the CompactRIO device.

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 VI you want to deploy to the CompactRIO device 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 from the host computer.

Building and Deploying a Stand-Alone Application

Use the **Real-Time Application Properties** dialog box, available by right-clicking **Build Specifications** under the **RT CompactRIO Target** in the **Project Explorer** window and selecting **New»Real-Time Application** from the shortcut menu, to create a build specification with the settings for the stand-alone application. Right-click the build specification in the **Project Explorer** window and select **Build** from the shortcut menu to build the application and deploy it to the CompactRIO device. Properties, settings, and built applications are downloaded to disk on the CompactRIO device.

You can choose to run a stand-alone application on the CompactRIO device at start-up. Right-click the build specification in the **Project Explorer** window and select **Run as startup** from the shortcut menu to build the application, deploy it to the CompactRIO device, and specify that it runs every time you start the CompactRIO device. Reboot the CompactRIO device to run the application.

Refer to the *Creating a Stand-Alone FRC Application* section of Chapter 5, *Tutorial: Creating an FRC cRIO Robot Project*, for more information about building and deploying a stand-alone application.

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 process images you acquire with the camera on the robot.

Use the **Create New FRC Dashboard Project** dialog box, available by clicking the **FRC Dashboard Project** link in the **Getting Started** window, to create an FRC dashboard project.

The FRC dashboard project looks similar to the following figure.

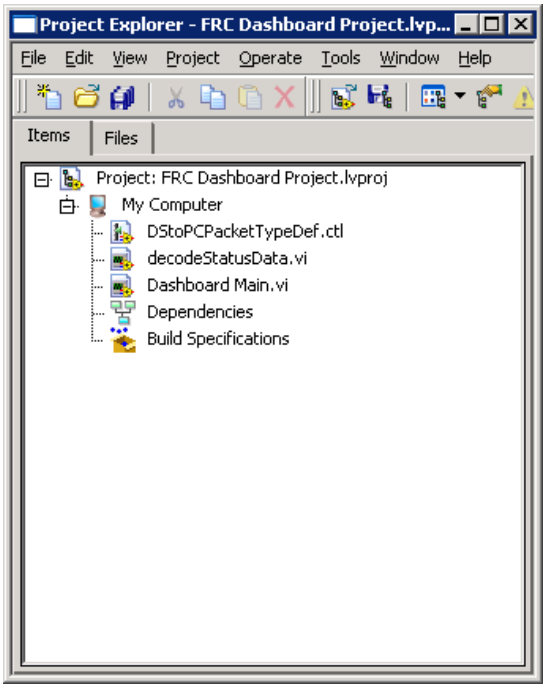


Figure 4-2. New FRC Dashboard Project

Whereas the FRC cRIO 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 two top-level VIs, the decodeStatusData VI and the Dashboard Main VI.

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. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. 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 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 about the robot.

5

Tutorial: Creating an FRC cRIO Robot Project

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

Creating an FRC cRIO Robot Project

Complete the following steps to create an FRC cRIO robot project.

1. Click the **FRC cRIO Robot Project** link in the **Getting Started Window** to display the **Create New FRC cRIO Robot Project** dialog box.
2. In the **Project name** text box, enter the name you want to use to identify the new FRC cRIO 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.`*`x`*`.`*`y`*`.2`, where *y* corresponds to the last two digits of the team number and *x* 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. Click the **Finish** button to close the **Create New FRC cRIO Robot Project** dialog box and create the new FRC cRIO robot project.

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

Deploying the FRC cRIO Robot Project

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

Modifying the Teleop VI

Complete the following steps to modify the Teleop VI to perform tank driving with two joysticks and Victor motor controllers.

1. Double-click the **Teleop.vi** item in the **Project Explorer** window to open the Teleop 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 to select the Init case.
4. Set the **RightMotorInverted** and **LeftMotorInverted** Boolean controls to TRUE or FALSE, depending on the orientation and wiring of the motors.
5. If the **Functions** palette is not visible on the block diagram, select **View»Functions Palette**.
6. Place an Open VI from the **Joystick** palette on the block diagram, below the existing Open VI.
7. Right-click the **JoystickDevice** input of the second Open VI and select **Create»Constant** from the shortcut menu.
8. From the **JoystickDevice** enum constant, select `2` to use the joystick connected to port 2 of the driver station.
9. Wire the **JoystickDevRef** output of the Open VI to the right border of the While Loop.

10. Right-click the output tunnel of the While Loop corresponding to the **JoystickDevRef** output and select **Replace with Shift Register** from the shortcut menu. Notice that the tunnel changes to a shift register, and a corresponding shift register is added to the left border of the While Loop.
11. Wire the shift register on the left border of the While Loop to the left border of the Case structure.
12. The block diagram should appear similar to the following figure.

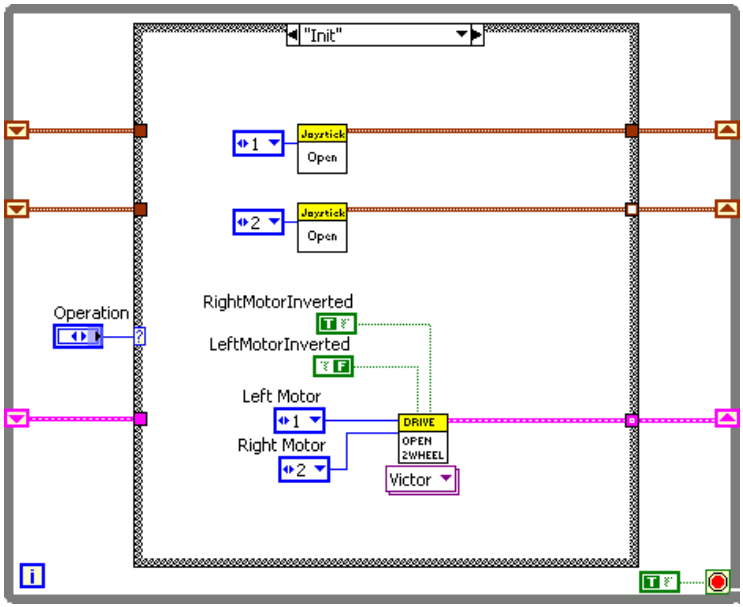


Figure 5-1. Modifying the Init Case of the Teleop VI

13. Click the increment or decrement arrows of the Case structure to select the Execute case.
14. Delete all code inside the Execute case.
15. Place a TankDriveJoystick VI on the block diagram.
16. Wire the input tunnel of the Case structure corresponding to the first joystick to the corresponding output tunnel of the Case structure.
17. Wire the same input tunnel to the **LeftStick** input of the TankDriveJoystick VI.
18. Wire the input tunnel of the Case structure corresponding to the second joystick to the corresponding output tunnel of the Case structure.

19. Wire the same input tunnel to the **RightStick** input of the TankDriveJoystick VI.
20. Wire the input tunnel of the Case structure corresponding to the RobotDriveDevRef to the **RobotDriveDevRef** input of the TankDriveJoystick VI.
21. Wire the **RobotDriveDevRef** output of the TankDriveJoystick VI to the corresponding output tunnel of the Case structure. The block diagram should appear similar to the following figure.

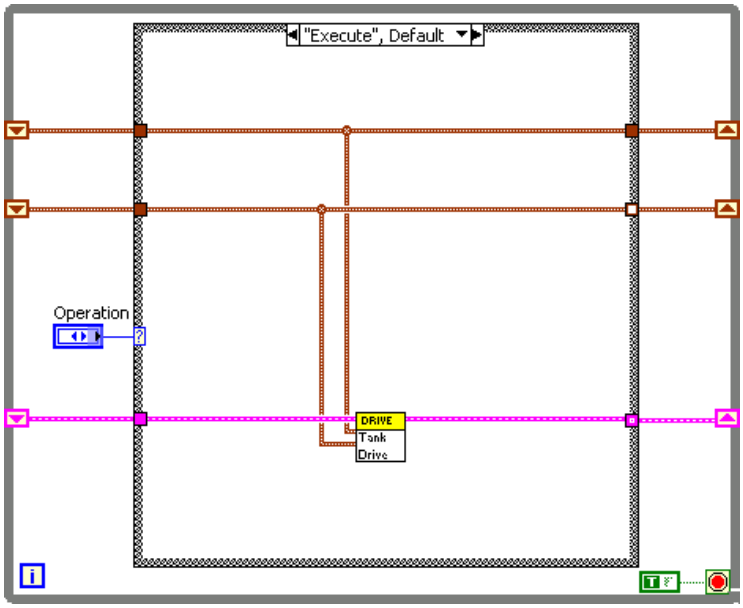


Figure 5-2. Modifying the Execute Case of the Teleop VI

22. Click the increment or decrement arrows of the Case structure to select the Stop case.

23. Wire the input tunnel of the Case structure corresponding to the second joystick to the corresponding output tunnel of the Case structure. The block diagram should appear similar to the following figure.

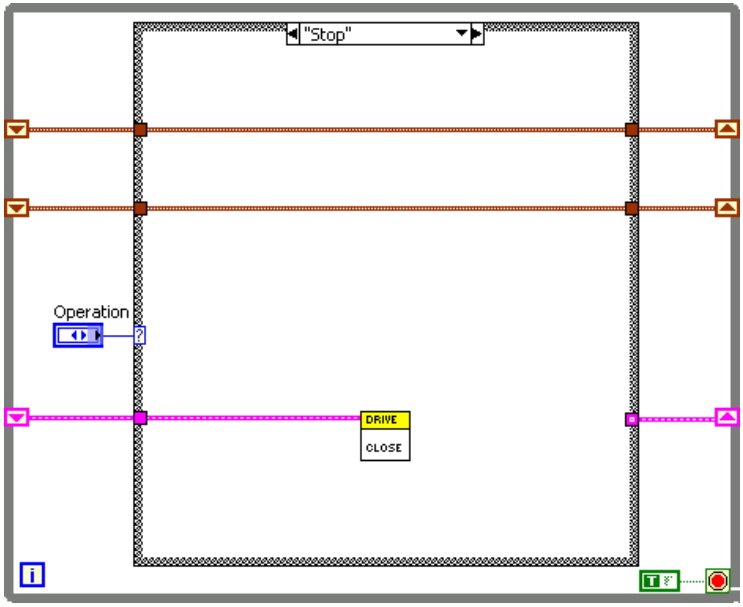


Figure 5-3. Modifying the Stop Case of the Teleop VI

The Teleop VI now allows you to perform tank driving. However, this VI cannot communicate with the joysticks or the motor controller from the host computer. You must deploy the FRC cRIO robot project that contains the Teleop VI to the CompactRIO device. You then can run the Robot Main VI, which calls the Teleop VI, on the CompactRIO device and communicate with the joysticks and motor controllers connected to the CompactRIO device. The following section, *Creating a Stand-Alone FRC Application*, describes how to deploy the FRC cRIO robot project to the CompactRIO device as a stand-alone application.

Running the FRC cRIO Robot Project

You can run the FRC cRIO robot project on the CompactRIO device and maintain a connection with the host computer to perform live front panel debugging.

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

1. In the **Project Explorer** window, double-click the **Robot Main.vi** item to open the Robot Main VI.
2. 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.
3. Move the joysticks and observe how the robot responds.
4. Click the **Stop** button of the 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.

Creating a Stand-Alone FRC Application

You can build the FRC cRIO 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 cRIO 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 **Build Specifications** under the **RT CompactRIO Target** and select **New»Real-Time Application** from the shortcut menu 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. From the **Project Files** tree on the **Source Files** page, select **Robot Main.vi**.
8. Click the **Add Item** arrow button next to the **Startup VIs** listbox to move the Robot Main VI to the **Startup VIs** listbox.
9. Select **Additional Exclusions** from the **Category** list.
10. On the **Additional Exclusions** page, remove the checkmark from the **Modify project library file after removing unused members** checkbox.
11. Click the **OK** button to close the **Real-Time Application Properties** dialog box.
12. In the **Project Explorer** window, right-click the build specification you created under the **RT CompactRIO Target** and select **Build** from the shortcut menu to build the application.
13. 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.
14. 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.

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

6

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 7 through 36 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.

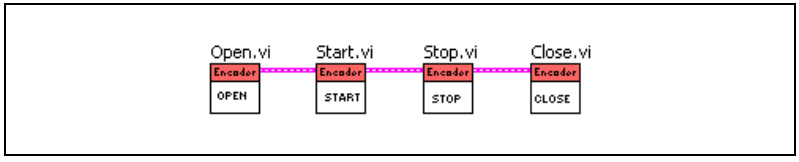


Figure 6-1. Using Reference Clusters with the Encoder VIs

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 as an **EncoderIndex** control.

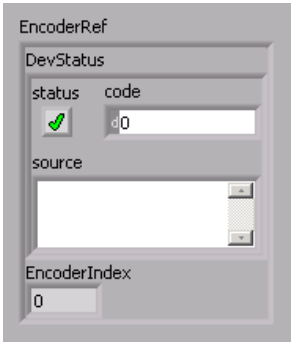


Figure 6-2. EncoderDevRef Input Reference Cluster

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 6-1, *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.

7

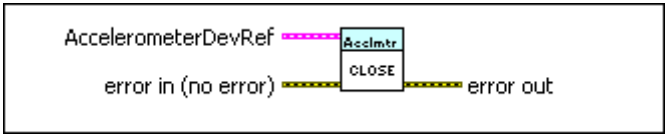
Accelerometer VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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 typically provided in the *FIRST* Robotics Competition (FRC) kit of parts is a two-axis accelerometer. The accelerometer provides acceleration data in the x- and y-axis relative to the circuit board. The accelerometer also can be used as a tilt sensor, actually measuring the acceleration of gravity.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

AI Module specifies the slot number on the CompactRIO device of the analog module you want to use. **AI Module** can specify a value of **1** or **2**. If **AI Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

AI Channel specifies the channel of the **AI Module** to which the accelerometer is connected. **AI Channel** can specify a value between 1 and 8. If **AI 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

error out passes error or warning information out of a VI to be used by other VIs. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

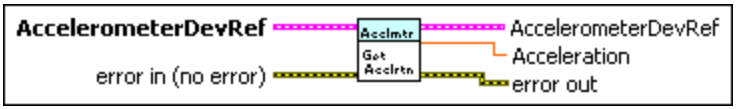
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

AI Module specifies the slot number on the CompactRIO device of the analog module you want to use. **AI Module** can specify a value of **1** or **2**. If **AI Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

AI Channel specifies the channel of the **AI Module** to which the accelerometer is connected. **AI Channel** can specify a value between 1 and 8. If **AI 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

AI Module returns the slot number of the module on the CompactRIO device.

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

Scaling returns the sensitivity of the accelerometer.

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

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

Acceleration returns the acceleration in floating point units of gravities.

error out passes error or warning information out of a VI to be used by other VIs. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

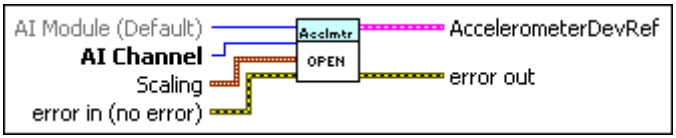
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Open.vi

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



AI Module (Default) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **Default** to specify the slot of the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default. You also can select **1** or **2** as the slot you want to use.

AI Channel specifies the channel of the **AI Module** that you want to use. Select a value between 1 and 8. If **AI 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

AI Module returns the slot number of the module on the CompactRIO device.

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

Scaling returns the sensitivity of the accelerometer.

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

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

error out passes error or warning information out of a VI to be used by other VIs. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

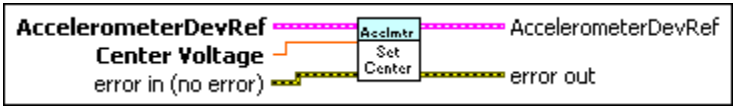
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

AI Module specifies the slot number on the CompactRIO device of the analog module you want to use. **AI Module** can specify a value of **1** or **2**. If **AI Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

AI Channel specifies the channel of the **AI Module** to which the accelerometer is connected. **AI Channel** can specify a value between 1 and 8. If **AI 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

AI Module returns the slot number of the module on the CompactRIO device.

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

Scaling returns the sensitivity of the accelerometer.

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

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

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

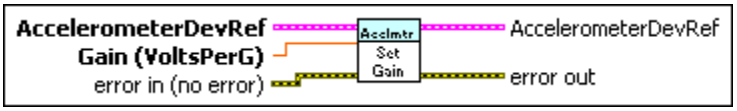
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

AI Module specifies the slot number on the CompactRIO device of the analog module you want to use. **AI Module** can specify a value of **1** or **2**. If **AI Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

AI Channel specifies the channel of the **AI Module** to which the accelerometer is connected. **AI Channel** can specify a value between 1 and 8. If **AI 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

AI Module returns the slot number of the module on the CompactRIO device.

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

Scaling returns the sensitivity of the accelerometer.

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

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

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

8

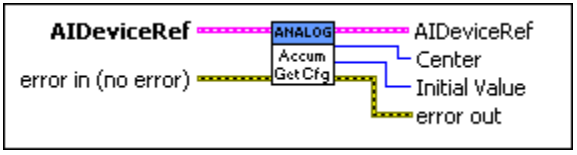
Accumulator VIs

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

Use the Accumulator VIs to configure and access the accumulator on the FPGA 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.

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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

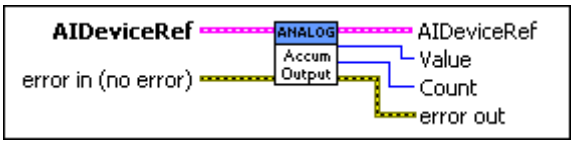
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

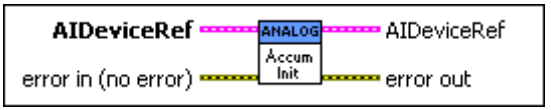
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

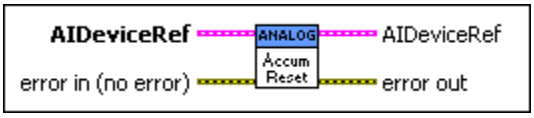
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

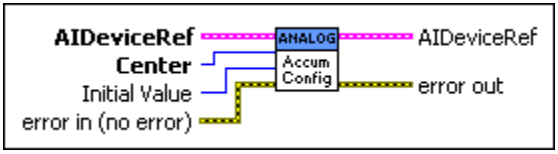
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

Channel specifies the channel of the **Analog Module** you want to use. **Channel** can specify a value between 1 and 8. If **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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

9

Actuators VIs

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

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

10

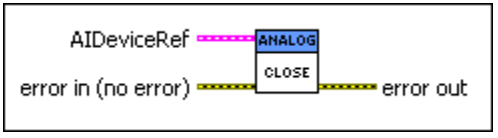
AnalogChannel VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

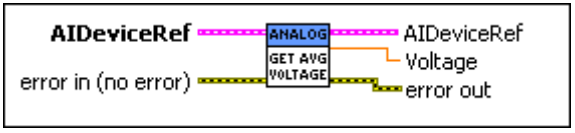
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default

analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

Voltage returns the voltage from the A/D converter.

error out passes error or warning information out of a VI to be used by other VIs. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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

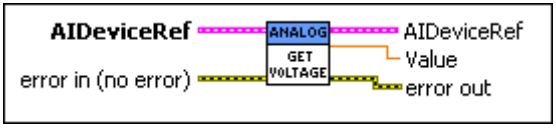
code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

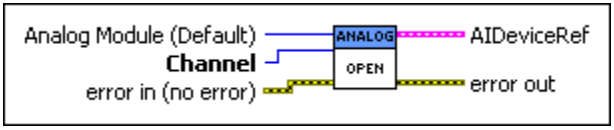
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **1** or **2**. You also can select **Default** to specify the slot of the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

11

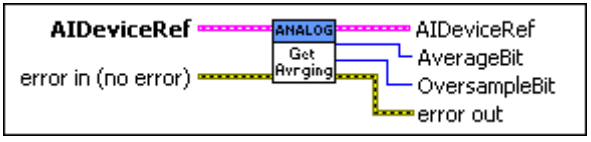
AnalogChannel Advanced VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

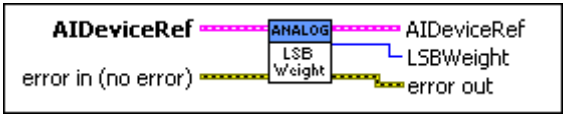
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

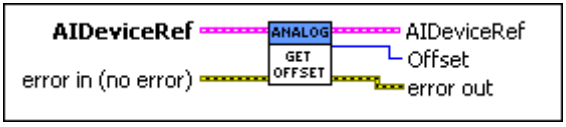
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

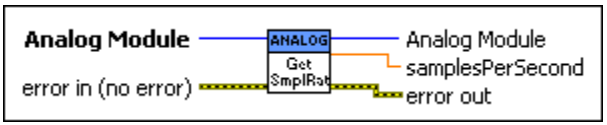
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 specifies the slot number on the CompactRIO device of the analog module you want to use. **Analog Module** can specify a value of **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

Analog Module returns the slot number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

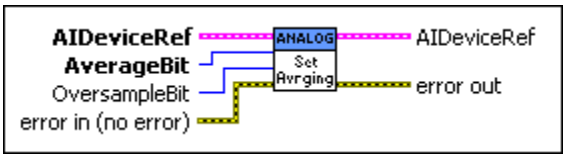
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

AverageBit specifies the number of bits to use when computing the average of a signal.

OversampleBit specifies the number of bits to use for oversampling the signal. The number of oversample values is 2^*n*, where *n* is **OversampleBit**.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

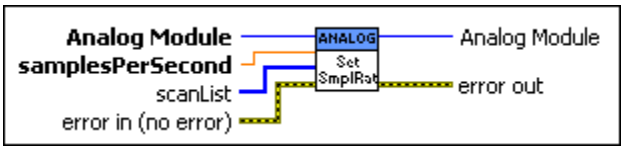
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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.



AIDeviceRef specifies a reference to the analog channel you want to use. Use the AnalogChannel Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

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

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

12

Analog Trigger VIs

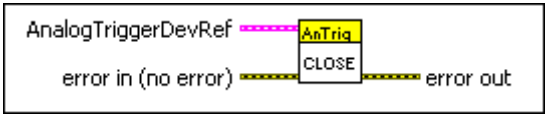
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Analog Trigger VIs.

Use the Analog Trigger 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 GetOutput 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.

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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

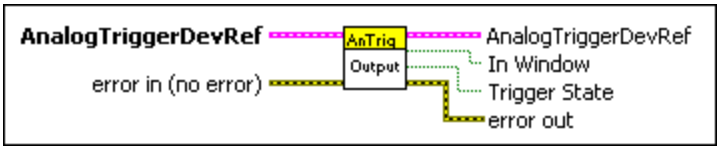
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **AnalogTriggerIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

AnalogTriggerIndex returns the index of the reserved analog trigger. **AnalogTriggerIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

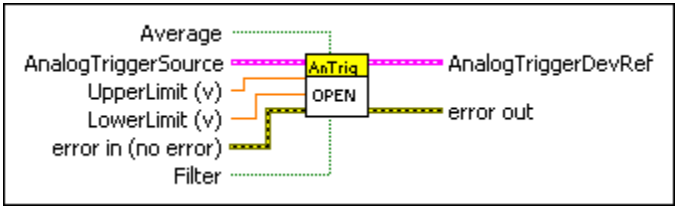
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Analog Trigger VIs.



Averaged specifies, when TRUE, that the analog source of the trigger contains averaged values.

AIDeviceRef specifies a reference to the analog channel you want to use. Use the Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **1** or **2**. If **Analog Module** specifies a value of **Default**, this VI uses the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

Channel specifies the channel of the **Analog Module** you want to use. **Channel** can specify a value between 1 and 8. If **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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

Filter specifies, when TRUE, that the analog source of the trigger contains filtered values.

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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

AnalogTriggerIndex returns the index of the reserved analog trigger. **AnalogTriggerIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

13

Camera VIs

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

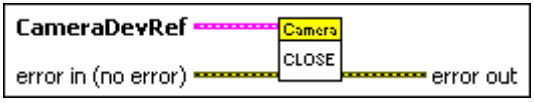
Use the Camera VIs to acquire images with the Axis camera. You then can use the *FIRST* Vision VIs in the *FIRST* Robotics Competition (FRC) cRIO robot project on the CompactRIO device to analyze and manipulate the images.

You use most of the Camera VIs in the FRC cRIO 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

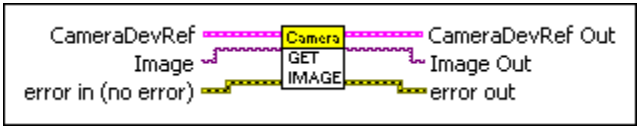
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 cRIO 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.



CameraDevRef specifies a reference to the camera you want to use. Use the Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 cRIO 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.

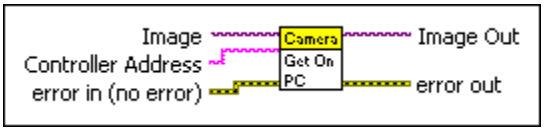


Image specifies a reference into which this VI copies the image it retrieves.

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.x.y.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.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

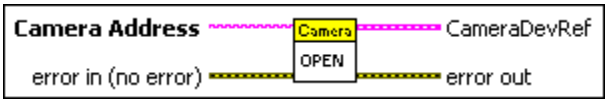
source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 cRIO 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.

CameraDevRef specifies a reference to the camera you want to use. Use the Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

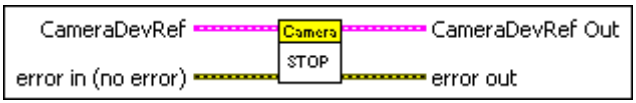
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

14

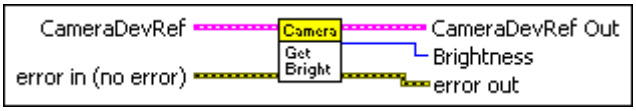
Camera Properties VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

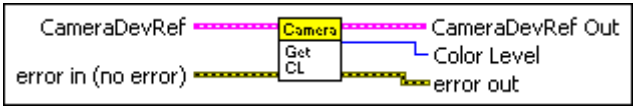
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select

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

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

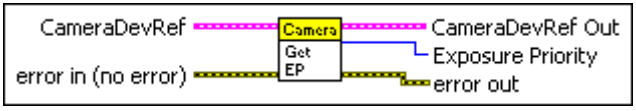
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

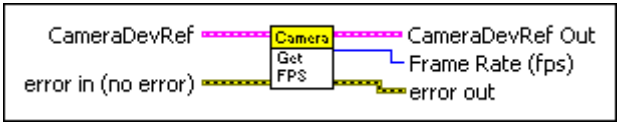
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

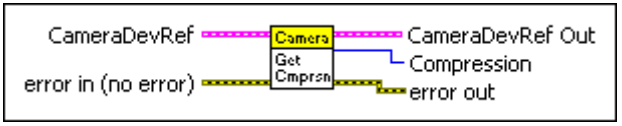
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Get Image Compression.vi

Returns the level of compression of the image you specify. 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

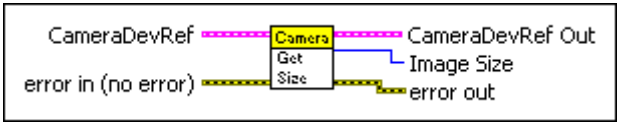
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

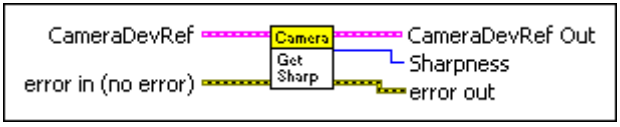
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

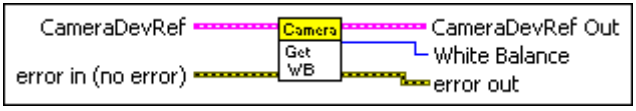
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

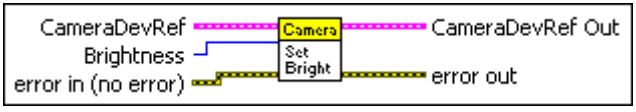
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

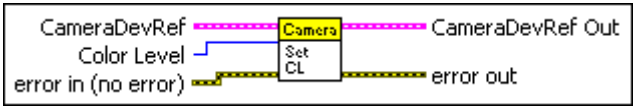
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

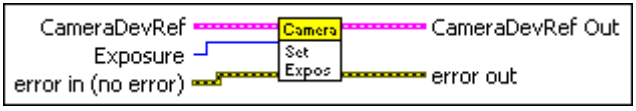
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

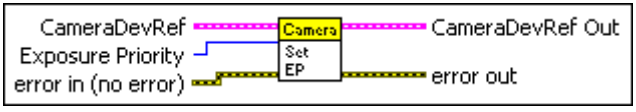
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

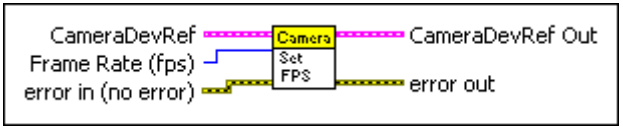
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

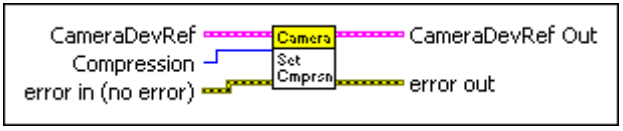
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Set Image Compression.vi

Sets the level of compression of the image you specify. 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

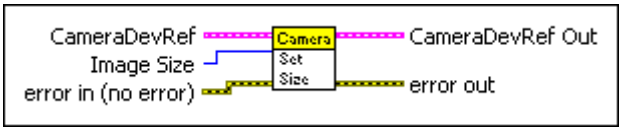
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

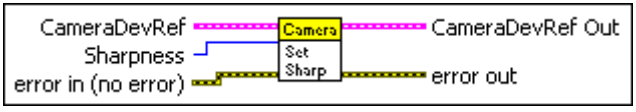
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

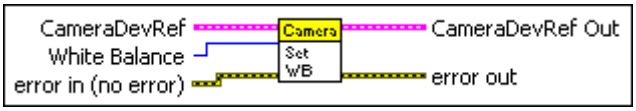
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Camera Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

15

Compressor VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

ThreadComm 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.

TaskVI specifies a reference to the RunCompressor VI.

Enabled specifies the run status of the compressor. TRUE specifies that the compressor is running, and FALSE specifies that it is idle.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

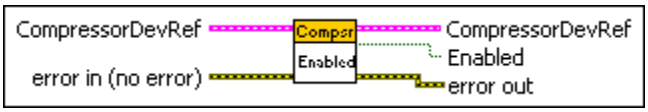
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

GetEnableState.vi

Returns the current state of the compressor as a Boolean indicator. TRUE indicates that the compressor is running, and FALSE indicates that the compressor is idle.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

ThreadComm 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.

TaskVI specifies a reference to the RunCompressor VI.

Enabled specifies the run status of the compressor. TRUE specifies that the compressor is running, and FALSE specifies that it is idle.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

ThreadComm returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

TaskVI returns a reference to the RunCompressor VI.

Enabled returns the run status of the compressor. TRUE indicates that the compressor is running, and FALSE indicates that it is idle.

Enabled returns the run status of the compressor. TRUE indicates that the compressor is running, and FALSE indicates that it is idle.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

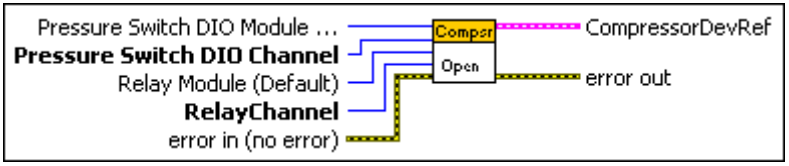
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the digital module on the CompactRIO device to which the pressure switch is connected. **Pressure Switch DIO Module (Default)** can specify a value of **4** or **6**. If **Pressure Switch DIO Module (Default)** 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, unless you specify a different digital module as the default.

Pressure Switch DIO Channel specifies the channel on the **Pressure Switch DIO Module** to which the pressure switch is connected. **Pressure Switch DIO Channel** can specify a value between 1 and 14. If **Pressure Switch DIO Channel** is **Invalid**, this VI returns an error.

Relay Module (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. **Relay Module (Default)** can specify a value of **4** or **6**. If **Relay Module (Default)** 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, unless you specify a different digital module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

ThreadComm returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

TaskVI returns a reference to the RunCompressor VI.

Enabled returns the run status of the compressor. TRUE indicates that the compressor is running, and FALSE indicates that it is idle.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

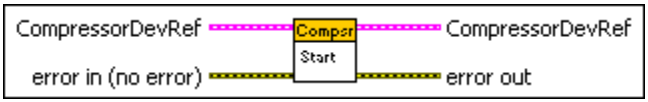
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

ThreadComm 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.

TaskVI specifies a reference to the RunCompressor VI.

Enabled specifies the run status of the compressor. TRUE specifies that the compressor is running, and FALSE specifies that it is idle.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

ThreadComm returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

TaskVI returns a reference to the RunCompressor VI.

Enabled returns the run status of the compressor. TRUE indicates that the compressor is running, and FALSE indicates that it is idle.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

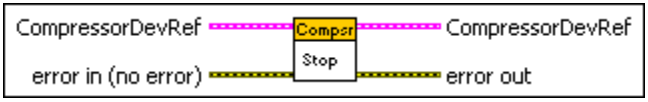
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

ThreadComm 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.

TaskVI specifies a reference to the RunCompressor VI.

Enabled specifies the run status of the compressor. TRUE specifies that the compressor is running, and FALSE specifies that it is idle.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

ThreadComm returns a reference to the FIFO. This parameter acts as a real-time status indicator of the compressor.

TaskVI returns a reference to the RunCompressor VI.

Enabled returns the run status of the compressor. TRUE indicates that the compressor is running, and FALSE indicates that it is idle.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

16

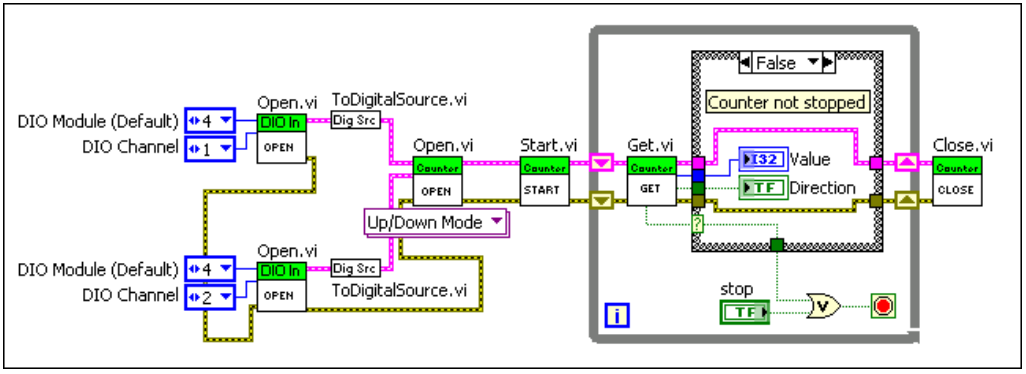
Counter VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

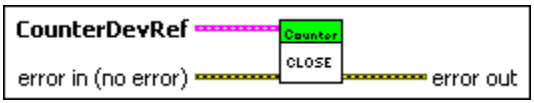
The following figure demonstrates how to return the count and direction of a counter.



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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

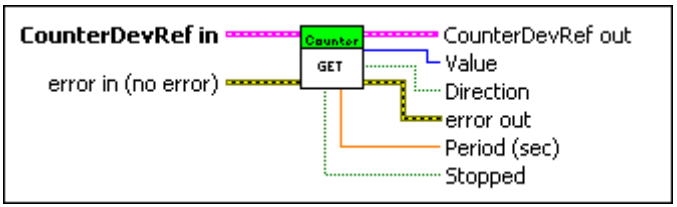
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

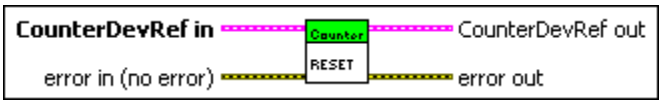
source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Period (sec) returns the 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 SetMaxPeriod 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

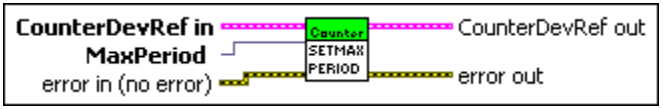
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SetMaxPeriod.vi

Specifies the maximum time between pulses that can elapse before the application considers the counter stopped. 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 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.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

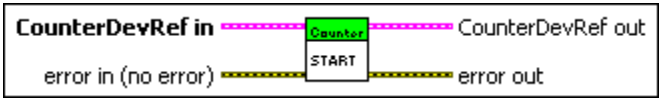
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

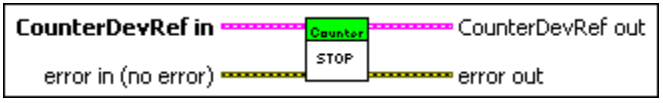
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

CntIndex returns the index of the reserved counter. **CntIndex** can return a value between 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

17

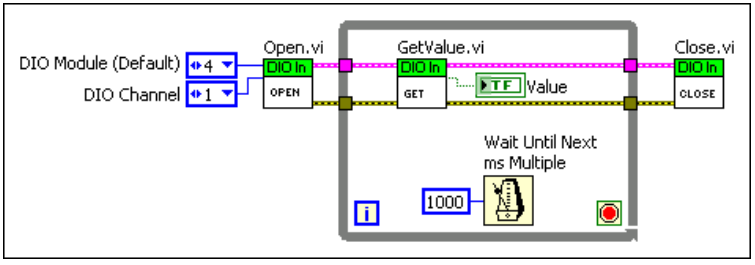
DigitalInput VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

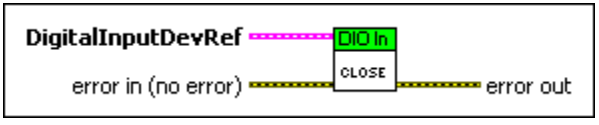
You can connect a digital line to either a digital sidecar or directly to a digital module on the CompactRIO device. A digital line can accept input signals or send output signals. Digital inputs accept signals from sensors, such as limit switches or touch sensors, whose signal has a value of either 0 or 1.

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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

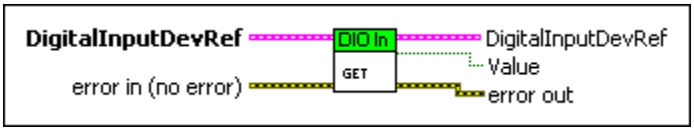
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

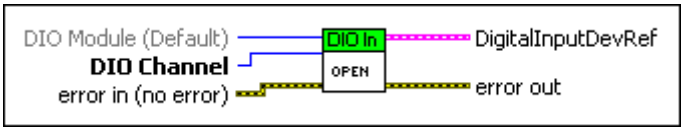
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

DIO Channel specifies the channel of the **DIO Module** that you want to use. Select a value between 1 and 14. If **DIO Channel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

18

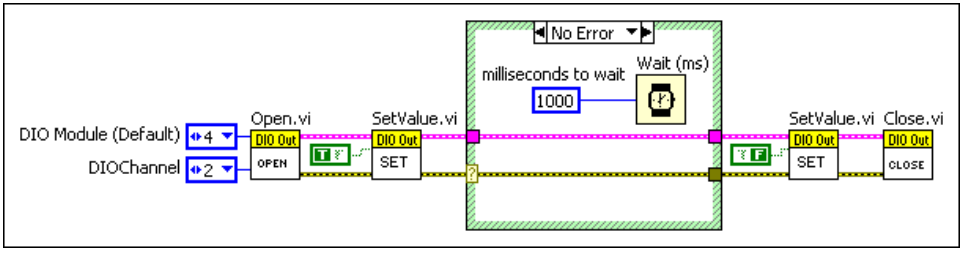
DigitalOutput VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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.

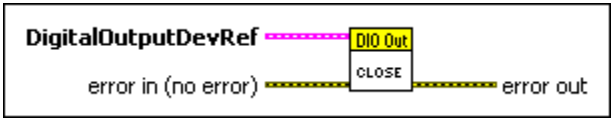


In the previous figure, the SetValue VI sets the digital output to TRUE for the duration of the pulse, which the Wait (ms) function specifies. The SetValue VI then sets the digital output back to FALSE at the end of the pulse. In this example, the pulse is software-timed. Alternatively, you can use the Pulse VI on the **Digital Output Advanced** palette 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

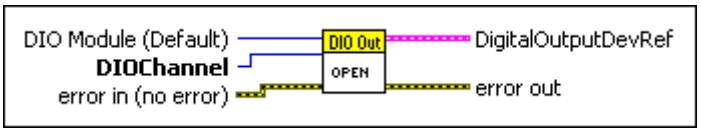
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

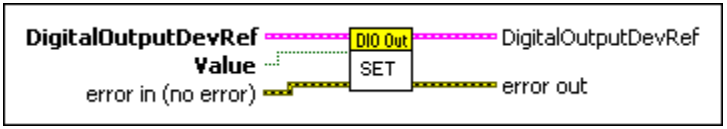
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

19

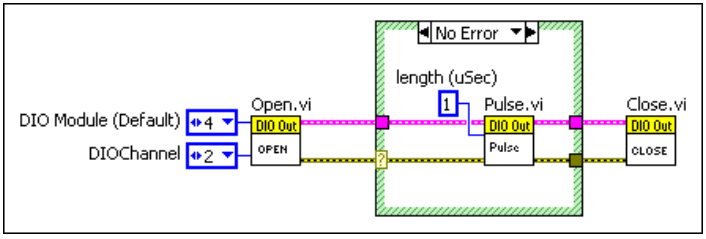
Digital Output Advanced VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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.

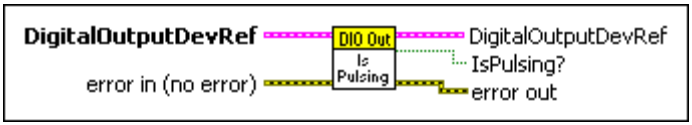


In the previous figure, the Pulse VI sets the digital output to the opposite of the current value for the duration you specify with the **length (uSec)** input. The Pulse VI emits a hardware-timed pulse. Alternatively, you can use the SetValue VI on the **DigitalOutput** palette and software timing to 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

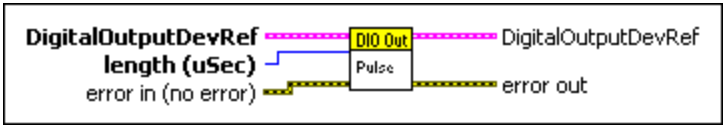
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 microseconds, 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

DIOChannel specifies the channel of the **DIO Module** that you want to use. **DIOChannel** can specify a value between 1 and 14. If **DIOChannel** 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

DIOChannel returns the channel of the **DIO Module** that you want to use. **DIOChannel** can return a value between 1 and 14. If **DIOChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

20

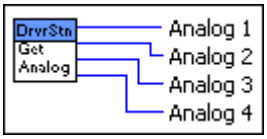
DriverStation VIs

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

Use the DriverStation VIs to manipulate data flow between the robot, driver station, and laptop. The DriverStation VIs can receive, interpret, and display both analog and digital data from your various sensors and joysticks. The DriverStation VIs accept signals from a laptop you connect to the driver station and from the robot. Use the DriverStation VIs to relay information from the laptop to the robot, check the battery power of the robot, set the controls of the robot based on the current competition mode, and display information from the outputs you define.

Get Analog Data.vi

Returns analog data from the driver station analog slots.



Analog 1 returns analog data from the driver station analog slot 1.

Analog 2 returns analog data from the driver station analog slot 2.

Analog 3 returns analog data from the driver station analog slot 3.

Analog 4 returns analog data from the driver station analog slot 4.

Get Competition Mode.vi

Returns the mode of the current phase of competition, which provides the user with the current restrictions of robot operation that apply. Options include **Autonomous Disabled**, **Autonomous Enabled**, **Teleop Disabled**, and **Teleop Enabled**.

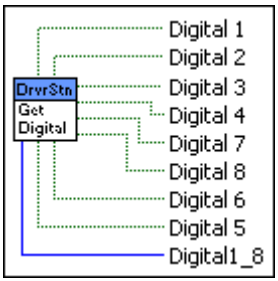
Robot Mode returns the allowed robot mode for the rules of the current phase of competition. Options include **Autonomous Disabled**, **Autonomous Enabled**, **Teleop Disabled**, and **Teleop Enabled**.

Derived Robot State returns the allowed derived robot state for the current phase of competition. Options include **Init**, **Execute**, and **Stop**.

RawControlByte returns the robot control byte in a raw, undeciphered state. In the communication between the driver station and the cRIO, a control byte exists that determines the state, mode, and special commands of the robot.

Get Digital Data.vi

Returns digital data from the driver station digital slots.



Digital 1 returns digital data from the driver station digital slot 1.

Digital 2 returns digital data from the driver station digital slot 2.

Digital 3 returns digital data from the driver station digital slot 3.

Digital 4 returns digital data from the driver station digital slot 4.

Digital 5 returns digital data from the driver station digital slot 5.

Digital 6 returns digital data from the driver station digital slot 6.

Digital 7 returns digital data from the driver station digital slot 7.

Digital 8 returns digital data from the driver station digital slot 8.

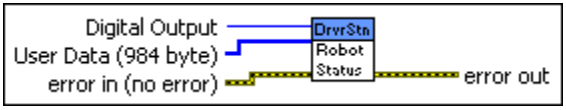
Digital1_8 encodes a combination of all of the Boolean digital data indicators.

Get Laptop Data.vi

Returns user-defined data that the VI receives from the laptop before sending the data to the robot.

Set Driver Station Status.vi

Defines what data is sent between the robot, the driver station, and the laptop.



Digital Output sets digital outputs on the driver station.

User Data (984 byte) specifies what data from user-created clusters on the laptop are sent to the driver station.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

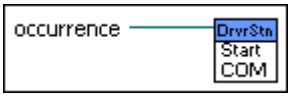
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Start Communication.vi

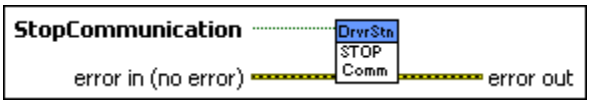
Starts the communication loop between the CompactRIO device and the driver station and initializes the host computer, joystick, and driver station data caches. This VI runs continuously and regularly checks for information from the host computer, joystick, and driver station. This VI also initializes the system watchdog on the CompactRIO device.



occurrence pauses the operation of the Robot Main VI in the FRC cRIO 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.

Stop Communication.vi

Stops the communication loop between the CompactRIO device and the driver station and closes the host computer, joystick, and driver station data caches.



StopCommunication stops the communication loop between the CompactRIO device and the driver station and closes the host computer, joystick, and driver station data caches.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

21

Encoder VIs

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

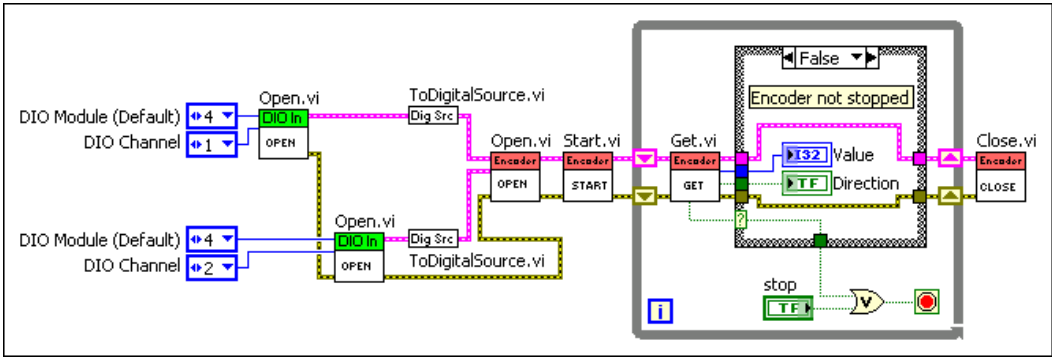
Use the Encoder VIs to get information about the rotation of a gear or motor, as measured by a quadrature encoder.

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 eight quadrature decoders that perform 4x decoding to interpret the information the encoders return. Whereas 1x decoding uses only the rising or falling edge of one digital signal to determine the count and direction of rotation, 4x decoding uses all rising and falling edges of both digital signals to determine this information.

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. The decoders are not tied to a specific module or channel on the CompactRIO device. You can reserve and release decoders as needed.

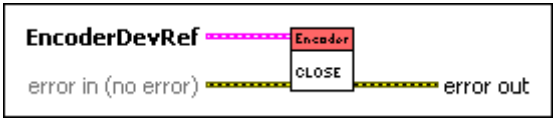
The following figure demonstrates how to return the count and direction of an encoder.



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 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 count 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **EncoderIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

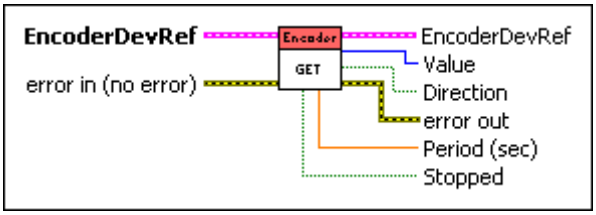
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7.
If **EncoderIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder.
EncoderIndex can return a value between 0 and 7.
If **EncoderIndex** returns a value of **`Invalid`**, the VI did not find the encoder you specified, and this VI returns an error.

Value returns the count of the encoder. The count corresponds to the number of edges in the digital signals that the encoder generates. The decoders on the CompactRIO device perform 4x decoding. Therefore, the number **Value** returns is four times greater than the number of full rotations of the encoder.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

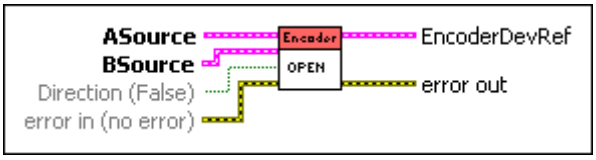
Period (sec) returns the period, in seconds, between pulses of the encoder.

Stopped returns TRUE if the encoder is stopped. The CompactRIO device considers an encoder 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 SetMaxPeriod 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.



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 ToDigitalSource VI to create this digital source.

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.

Module specifies the slot number on the CompactRIO device of the digital module you want to use. **Module** can specify a value of **4** or **6**. If **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, unless you specify a different digital module as the default.

DigitalChannel specifies the digital input channel of the **Digital Module** to which the encoder is connected. **DigitalChannel** can specify a value between 1 and 14. If **DigitalChannel** is **Disabled**, the device using the digital source always receives a value of 0 for the signal on that channel.

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 0 and 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 ToDigitalSource VI to create this digital source.

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.

Module specifies the slot number on the CompactRIO device of the digital module you want to use. **Module** can specify a value of **4** or **6**. If **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, unless you specify a different digital module as the default.

DigitalChannel specifies the digital input channel of the **Digital Module** to which the encoder is connected. **DigitalChannel** can specify a value between 1 and 14. If **DigitalChannel** is **Disabled**, the device using the digital source always receives a value of 0 for the signal on that channel.

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 0 and 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.

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. **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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between 0 and 7.
If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

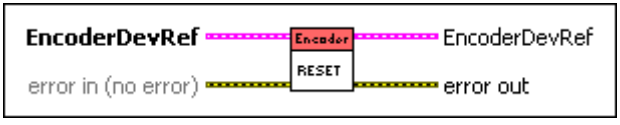
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **EncoderIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between 0 and 7. If **EncoderIndex** returns a value of **`Invalid`**, the VI did not find the encoder 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

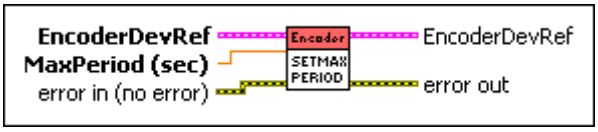
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SetMaxPeriod.vi

Specifies the maximum time between pulses that can elapse before the CompactRIO device considers the encoder to be stopped. 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **EncoderIndex** is **Invalid**, this VI returns an error.

MaxPeriod (sec) specifies the maximum time, in seconds, that can elapse between pulses before the CompactRIO device considers the encoder stopped. The default is 0.05. The CompactRIO device considers an encoder stopped if the CompactRIO device does not detect a new pulse and the number of counts does not change during the period you specify. Use the Get VI to return the stopped status of the encoder.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between 0 and 7. If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

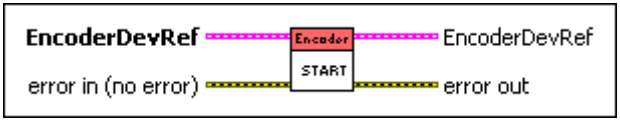
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **EncoderIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between 0 and 7.
If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

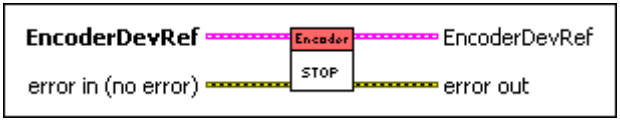
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 0 and 7. If **EncoderIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

EncoderIndex returns the index of the reserved encoder. **EncoderIndex** can return a value between 0 and 7.
If **EncoderIndex** returns a value of `Invalid`, the VI did not find the encoder 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

22

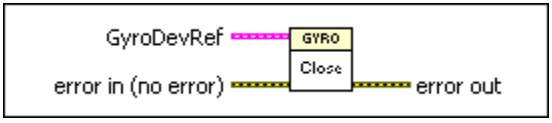
Gyro VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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. The Gyro VIs also 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Analog Module specifies the slot number of the module on the CompactRIO device.

Channel specifies the channel number of the module on the CompactRIO device.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Analog Module specifies the slot number of the module on the CompactRIO device.

Channel specifies the channel number of the module on the CompactRIO device.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

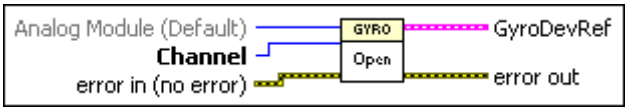
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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.



Analog Module (Default) specifies the slot number on the CompactRIO device of the analog module you want to use. Select **1** or **2**. You also can select **Default** to specify the slot of the default analog module. The default analog module is the first analog module, or the module in slot 1, unless you specify a different analog module as the default.

Channel specifies the channel of the **Analog Module** that you want to use. The default **Channel** value is **1**. If **Channel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Analog Module specifies the slot number of the module on the CompactRIO device.

Channel specifies the channel number of the module on the CompactRIO device.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Analog Module specifies the slot number of the module on the CompactRIO device.

Channel specifies the channel number of the module on the CompactRIO device.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Analog Module returns the slot number of the module on the CompactRIO device.

Channel returns the channel number of the module on the CompactRIO device.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

23

IO VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

24

Jaguar VIs

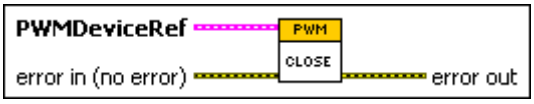
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Jaguar VIs.

Use the Jaguar VIs to set or get the speed of a motor using a Jaguar motor controller. If you use a Jaguar motor controller, use the Open VI on this palette instead of the Open VI on the PWM palette. The Jaguar Open VI uses preconfigured PWM range and deadband values specific to the Jaguar motor controller so you do not need to perform any conversions or scaling calculations on the raw PWM values.

Close.vi

Closes the reference to the PWM, motor controller, or servo you specify. Use this VI to close each PWM reference that you open with the Open VI of the motor controller or servo you use.

The same Close VI appears on the PWM, Jaguar, Victor, and Servo palettes. This Close VI closes references to different motor controller or servo Open VIs. However, you must use the specific motor controller or servo Open VI to open a reference to the device. For example, use the Open VI on the Jaguar palette to open a reference to the Jaguar motor controller.



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 PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

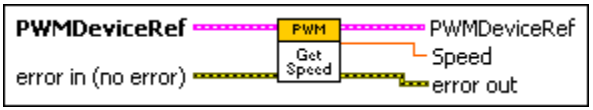
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

GetSpeed.vi

Returns the scaled value from the PWM you specify. The scaled value represents speed. If you use a Jaguar or Victor motor controller, 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.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other Victor VIs or Jaguar VIs, use the Open VI on the **Victor** or **Jaguar** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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. If you use a Jaguar or Victor motor controller, 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

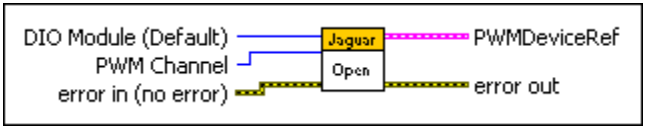
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Open.vi

Creates a reference to the Jaguar motor controller you specify. You must open a reference before using any other VIs on this palette.



DIO Module (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module you want to use. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

Eliminate Deadband, when TRUE, enables a set of scaling constants that eliminate the effects of the deadband in the motor controller. The default is FALSE.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 `4` or `6`. If **DIO Module** returns a value of `Default`, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

SetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

GetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

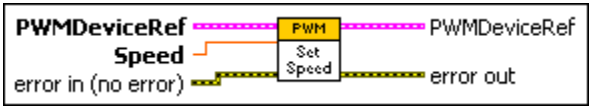
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other Victor VIs or Jaguar VIs, use the Open VI on the **Victor** or **Jaguar** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **6**. If **DIO Module** returns a value of **Default**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

25

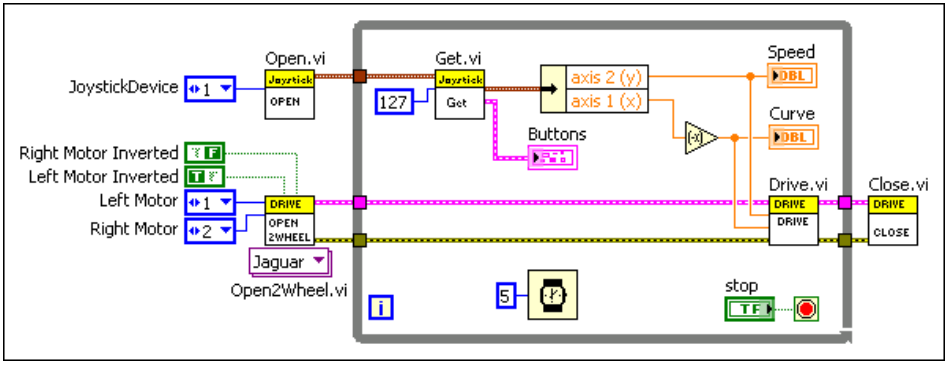
Joystick VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

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.

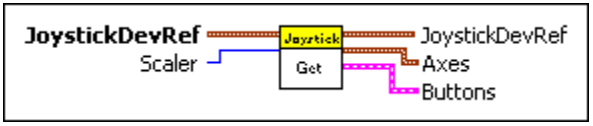
The following figure demonstrates how to open a reference to joystick 1, get the button values and scaled axis values for that joystick, and use those values to specify the **Speed** and **Curve** inputs for the Drive VI. Joystick 1 has resolution of 256 × 256, so you set the **Scaler** input of the Get VI to 127.



The RobotDrive VIs in the previous figure control a two-wheeled robot using Jaguar motor controllers. The Drive VI computes the motor speed from the **Speed** and **Curve** values you specify.

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.

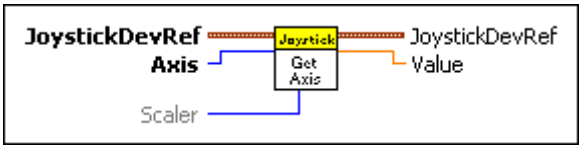
Button 10 returns TRUE when button 10 on the joystick is pressed.

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.

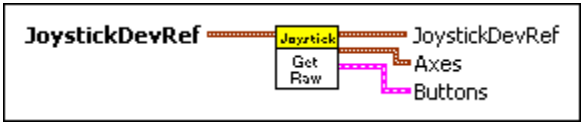
JoystickDevRef returns a reference to the joystick.

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.

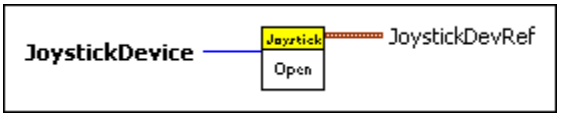
Button 10 returns TRUE when button 10 on the joystick is pressed.

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.

26

MotorControl VIs

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

Use the MotorControl VIs to control the speed and direction of motors and the position of servos.

27

PWM VIs

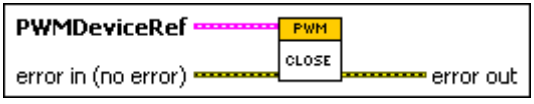
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the PWM VIs.

Use the PWM VIs to generate a hobby-style pulse width modulation (PWM) signal. You can use PWM signals to drive motor controllers and servos. Use the VIs on this palette with custom motor controllers or servos. Otherwise, use the VIs on the **Victor**, **Jaguar**, or **Servo** palettes to use PWM for those devices.

Close.vi

Closes the reference to the PWM, motor controller, or servo you specify. Use this VI to close each PWM reference that you open with the Open VI of the motor controller or servo you use.

The same Close VI appears on the PWM, Jaguar, Victor, and Servo palettes. This Close VI closes references to different motor controller or servo Open VIs. However, you must use the specific motor controller or servo Open VI to open a reference to the device. For example, use the Open VI on the **Jaguar** palette to open a reference to the Jaguar motor controller.



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 PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

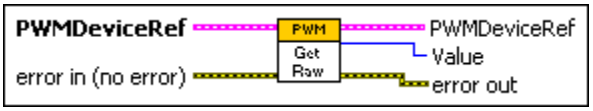
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 controllers or servos. Use the GetSpeed VI on the **Victor** or **Jaguar** palettes to return PWM values from the Jaguar or Victor controller.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

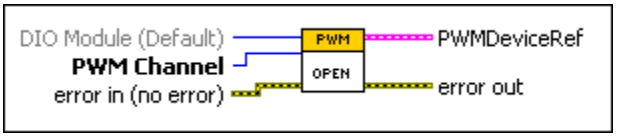
code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 controllers or servos. You must use the Open VI on the **Jaguar**, **Victor**, and **Servo** palettes to open references to those devices. For example, use the Open VI on the **Jaguar** palette to open a reference to the Jaguar motor controller.

By default, the center of the signal is 128.



DIO Module (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

PWM Channel specifies the PWM channel you want to use. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

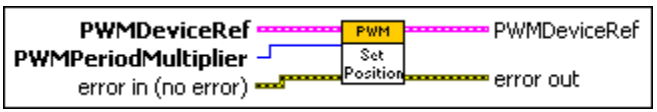
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **6**. If **DIO Module** returns a value of **Default**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

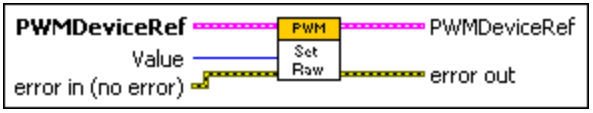
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 controllers or servos. Use the SetSpeed VI on the **Victor** or **Jaguar** palettes to specify PWM values for the Jaguar or Victor controller.



PWMDeviceRef specifies a reference to the PWM you want to use. Use the Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or

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

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 **4** or **6**. If **DIO Module** returns a value of **Default**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

28

Relay VIs

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

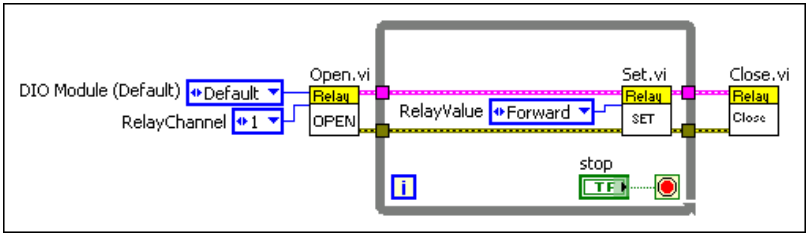
Use the Relay VIs to manipulate relays. You can open and close relays to control the power source for a motor or other device.

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.

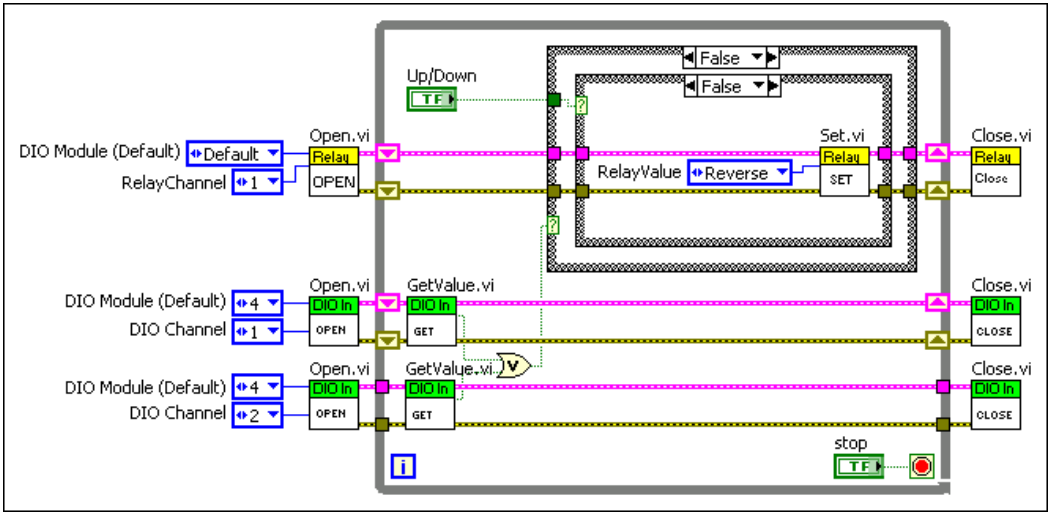
You also can use relays with a pressure switch and the Compressor VIs to maintain pressure with a compressor. When the pressure is low, the pressure switch can turn the relay on, which in turn can start the compressor.

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.

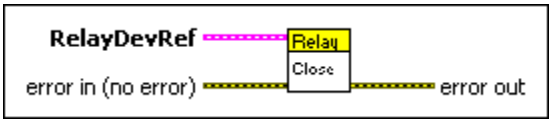


The following figure demonstrates how to use the Relay VIs and the DigitalInput VIs to control the direction of a motor and to stop the motor if either of two limit switches is triggered. A relay is connected to relay channel 1 of the default digital module on the CompactRIO device. Two digital input 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

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

RelayDirection specifies the direction in which a motor corresponding to the relay moves.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

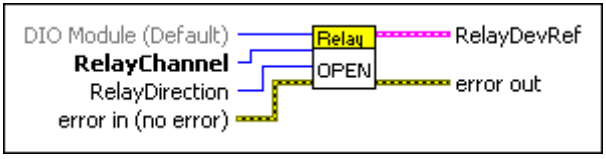
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

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

RelayDirection specifies the direction in which a motor corresponding to the relay moves.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or

function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

RelayChannel returns the channel of the digital module to which the relay is connected. **RelayChannel** can return a value between 1 and 8. If **RelayChannel** returns a value of **`Invalid`**, the VI did not find the relay you specified, and this VI returns an error.

RelayDirection 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

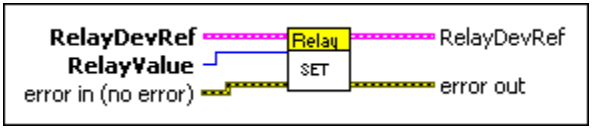
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

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

RelayDirection 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

RelayChannel returns the channel of the digital module to which the relay is connected. **RelayChannel** can return a value between 1 and 8. If **RelayChannel** returns a value of **`Invalid`**, the VI did not find the relay you specified, and this VI returns an error.

RelayDirection 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

29

RobotDrive VIs

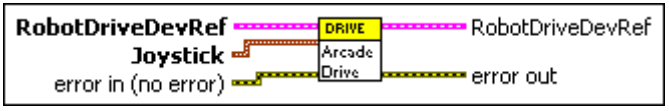
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the RobotDrive VIs.

The RobotDrive VIs provides VIs that you can use to control the drive motors on your robot. You have many options for designing and configuring the robot. The RobotDrive VIs support several configurations. The RobotDrive VIs work with two-wheel and four-wheel robots, holonomic and non-holonomic robots, joystick and autonomous control, and arcade and tank driving. You can choose three types of VIs: open, drive, and close.

The utility class for handling the driving of the robot is based on a definition of the motor configuration. The robot drive class handles the basic driving of the robot. RobotDrive VIs support 2 and 4 motor standard drive trains. Motor channel numbers are supplied on creation of the class. You use the motor channel numbers for either the Drive VI, intended for hand-created drive code, or with the TankDrive or ArcadeDrive VIs used for manually controlled driving.

ArcadeDriveJoystick.vi

Executes the Get VI from the Joystick VIs to get the values of the x-axis and y-axis joystick output signals. This VI then encodes the x-axis and y-axis values to supply different signals to each motor for turning. Arcade driving uses a single joystick where the y-axis motion controls driving speed and the x-axis motion controls turning rate.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

Joystick specifies the number of the port on the operator interface connected to the joystick.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function

runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of `4` or `6`. If **DIO Module** returns a value of `Default`, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

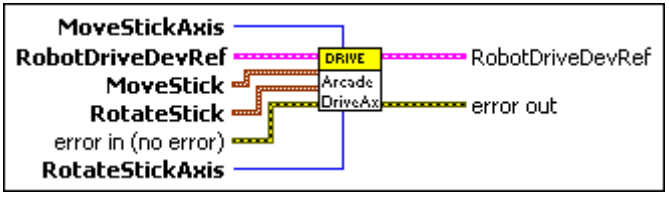
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

ArcadeDriveJoystickAxis.vi

Accepts inputs from two of four ports (1, 2, 3, 4). One input controls the move joystick and one controls the rotate joystick. This VI also defines the input mapping by specifying the axis for the move joystick and the axis for the rotate joystick.



MoveStickAxis specifies the axis on the joystick that controls the forward and backward motion of the robot. Select a value between Axis 1 and Axis 6.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

MoveStick specifies which joystick moves the robot forward and backward.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

RotateStick specifies which joystick rotates the robot around a center point.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

RotateStickAxis specifies the axis on the joystick that controls the turning motion of the robot. Select a value between **1** and **6**.

RobotDriveDevRef returns a reference to the robot drive you want to use.

DevStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

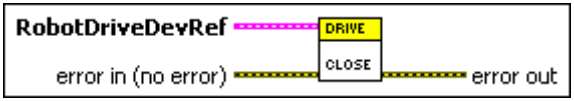
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

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 Open2Wheel VI or the Open4Wheel 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

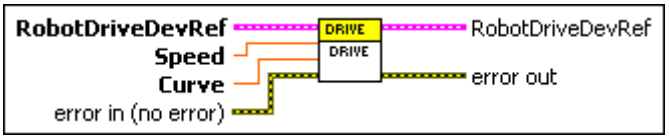
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Drive.vi

Controls a skid-steered robot by driving motors on one side of the robot at a different speed than the motors on the other side of the robot. This VI computes the left and right motor speed from two inputs, **Speed** and **Curve**, and sends command signals to the motors. The **Speed** and **Curve** values pass from a joystick or from autonomous driving.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

Curve sets the degree of the responsiveness of the robot to the movement of the joystick.

Speed sets the degree of the responsiveness of the robot to movement on the y-axis of the joystick.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

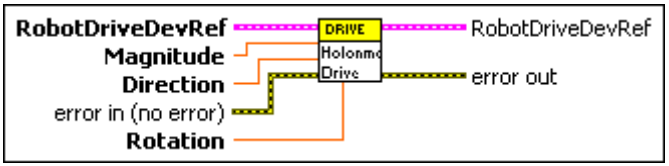
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

HolonomicDrive.vi

Controls robots with special wheels that allow the robot to drive in any direction without first rotating or maneuvering the robot. 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 that is perpendicular to the direction of travel of the main wheel without having to first move the main wheel. The robot can travel in any direction by moving the main wheel and the rollers at different speeds. This VI requires the robot to have four omni-directional wheels and four motors. This VI calculates the correct speeds for the wheels and rollers from **Magnitude**, **Direction**, and **Rotation** inputs. Each wheel must rotate at a particular rotational velocity and direction for a robot with omni-directional wheels to obtain and move along a particular heading.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

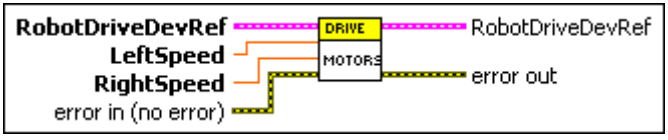
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

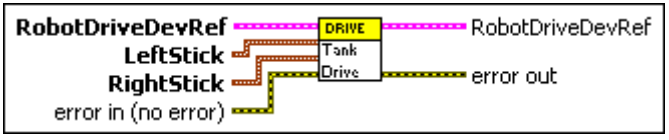
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

TankDriveJoystick.vi

Enables you to operate a robot with tank drive motor configuration. Tank driving uses two joysticks. You control the left and right motors independently with the two joysticks. This VI uses the two y-axis values from the joysticks to supply signals to each motor for robot movement.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM**

Channel can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

LeftStick specifies which joystick controls the left motor.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

RightStick specifies which joystick controls the right motor.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

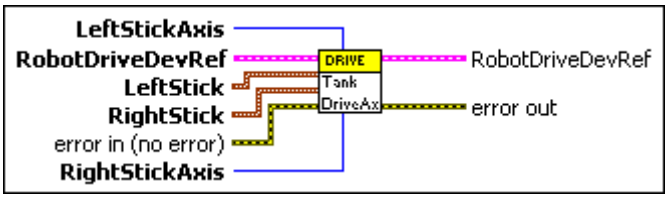
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

TankDriveJoystickAxis.vi

Enables you to choose the axis from either joystick to use for the motor speed. This VI defines the joystick data that the TankDriveJoystick VI uses.



LeftStickAxis specifies the axis of movement on which the left stick operates. Select an axis between Axis 1 and Axis 6.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

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

Motors[] specifies information about the motor you select.

Inverted? specifies the direction that the motor runs as either forward or reverse. **Inverted?** specifies FALSE for forward and TRUE for reverse.

PWMDeviceRef specifies a reference to the pulse-width modulation (PWM) you want to use. The PWM specifies the amount of power sent to the motors.

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

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

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies a pulse-width modulation (PWM) output channel on a digital sidecar that controls the motor. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

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.

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.

Sensitivity specifies the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? specifies that the PWM channel was allocated internally. TRUE destroys the PWM, and FALSE does not destroy the PWM.

LeftStick specifies which joystick controls the left motor.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

RightStick specifies which joystick controls the right motor.

Device specifies the number of the port on the operator interface connected to the joystick. Select a port value between 1 and 4.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

RightStickAxis specifies the axis of movement on which the right stick operates. Select an axis between Axis 1 and Axis 6.

RobotDriveDevRef returns a reference to the robot drive you want to use.

DevStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

Motors[] returns information about the motor you specify.

Inverted? returns the direction of the motor. **Inverted?** returns FALSE for forward and TRUE for reverse.

PWMDeviceRef returns a reference to the PWM you want to use. The PWM specifies the amount of power sent to the motors.

DeviceStatus can accept error information wired from VIs previously called. Use this information to decide if any functionality should be bypassed in the event of errors from other VIs.

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

DIO Module returns the slot number on the CompactRIO device of the digital module to which the relay is connected. **DIO Module** can return a value of **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the relay is connected to the default digital module. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns a PWM output channel on a digital sidecar that controls the motor.

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.

TransformRef returns a reference that allows teams to define a non-linear relationship to compensate for motor response.

InvTransformRef returns a reference to the converter that this VI uses to convert speed values to raw values.

Sensitivity returns the sensitivity of the joystick. The greater the sensitivity value of the joystick, the greater the responsiveness of the robot.

DestroyPWM? returns the status of the PWM channel. TRUE destroys the PWM, and FALSE does not destroy the PWM.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

30

Sensors VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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.

31

Servo VIs

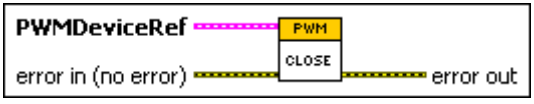
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Servo VIs.

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.

Close.vi

Closes the reference to the PWM, motor controller, or servo you specify. Use this VI to close each PWM reference that you open with the Open VI of the motor controller or servo you use.

The same Close VI appears on the PWM, Jaguar, Victor, and Servo palettes. This Close VI closes references to different motor controller or servo Open VIs. However, you must use the specific motor controller or servo Open VI to open a reference to the device. For example, use the Open VI on the **Jaguar** palette to open a reference to the Jaguar motor controller.



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 PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

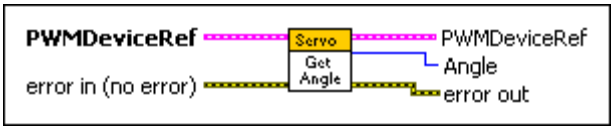
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Servo VIs, use the Open VI on the **Servo** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 `4` or `6`. If **DIO Module** returns a value of `Default`, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

SetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

GetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

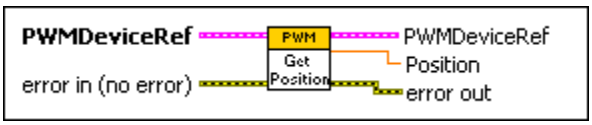
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Servo VIs, use the Open VI on the **Servo** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 `4` or `6`. If **DIO Module** returns a value of `Default`, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

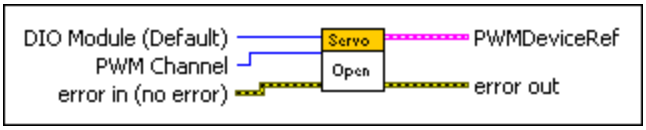
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module you want to use. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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

error in and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

SetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

GetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

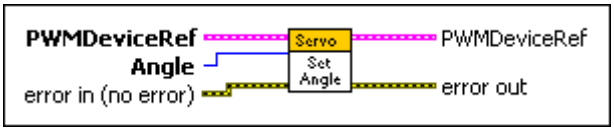
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Servo VIs, use the Open VI on the **Servo** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 `4` or `6`. If **DIO Module** returns a value of `Default`, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

SetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

GetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

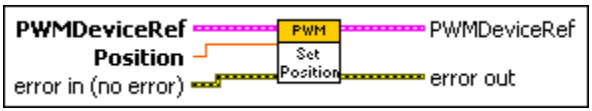
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Servo VIs, use the Open VI on the **Servo** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **6**. If **DIO Module** returns a value of **Default**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

SetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

GetSpeed is the scale factor the VI uses to convert raw PWM values to speed values. You do not need to edit this parameter.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

32

Solenoid VIs

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

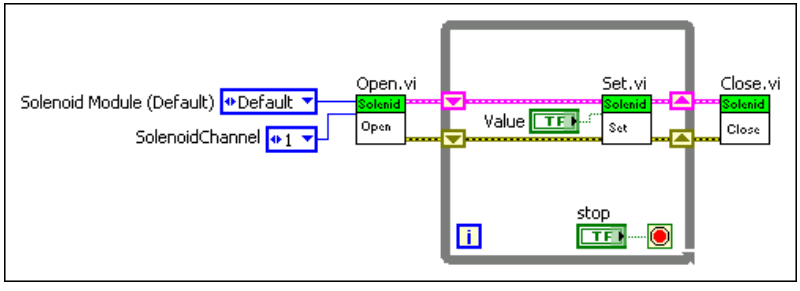
Use the Solenoid VIs to operate a solenoid. A solenoid is a pneumatic valve that can release air to move a part on the robot.

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.

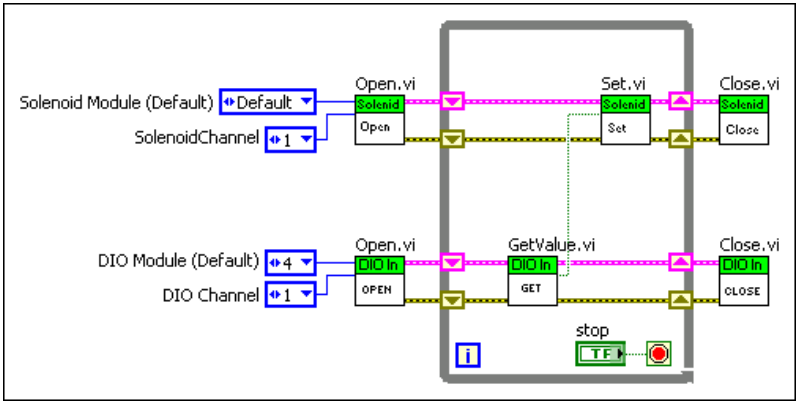
Each time the solenoid releases air, the air pressure of the compression system drops. If the air pressure drops below a certain level, you can use the Compressor VIs to turn on the pressure switch of the compressor. The compressor then pumps more air into the system, thus increasing the air pressure.

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.

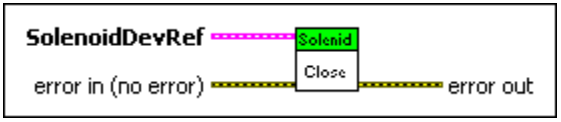


The following figure demonstrates how to set the value of a solenoid based on a digital input value. A solenoid is connected to channel 1 of the default solenoid module on the CompactRIO device. A digital line is connected to channel 1 of the digital module in slot 4 of the CompactRIO device. The GetValue 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can specify a value of **8**. If **Solenoid Module** specifies a value of **Default**, this VI uses the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

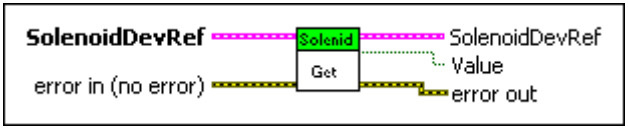
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can specify a value of **8**. If **Solenoid Module** specifies a value of **Default**, this VI uses the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can return a value of **`8`**. If **Solenoid Module** returns a value of **`Default`**, the solenoid is connected to the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

SolenoidChannel returns the channel of the digital module to which the solenoid is connected. **SolenoidChannel** can return a value between 1 and 8. If **SolenoidChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

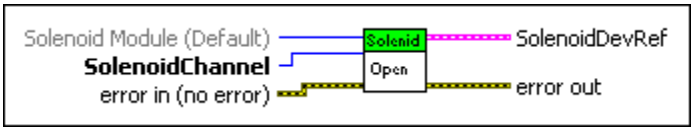
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 (Default) specifies the slot number on the CompactRIO device of the solenoid module you want to use. Select **8**. You also can select **Default** to specify the slot of the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

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

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

- **status** is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.
- **code** is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.
- **source** describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can return a value of `8`. If **Solenoid Module** returns a value of `Default`, the solenoid is connected to the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

SolenoidChannel returns the channel of the digital module to which the solenoid is connected. **SolenoidChannel** can return a value between 1 and 8. If **SolenoidChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

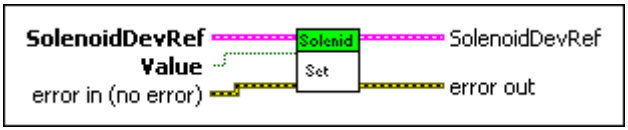
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can specify a value of **8**. If **Solenoid Module** specifies a value of **Default**, this VI uses the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

SolenoidChannel specifies the channel of the **Solenoid Module** that you want to use. **SolenoidChannel** can specify a value between 1 and 8. If **SolenoidChannel** 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** can return a value of `8`. If **Solenoid Module** returns a value of `Default`, the solenoid is connected to the default solenoid module. The default solenoid module is the first solenoid module, or the module in slot 8, unless you specify a different solenoid module as the default.

SolenoidChannel returns the channel of the digital module to which the solenoid is connected. **SolenoidChannel** can return a value between 1 and 8. If **SolenoidChannel** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

33

Ultrasonic VIs

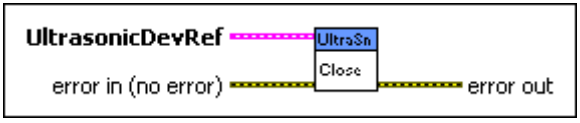
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Ultrasonic VIs.

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 object in front of it and returns to the sensor. The CompactRIO device uses a counter to calculate how long the echo of the ping takes to return to the sensor and uses this time to calculate how far away the object is. You can choose to measure the distance in inches or millimeters.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

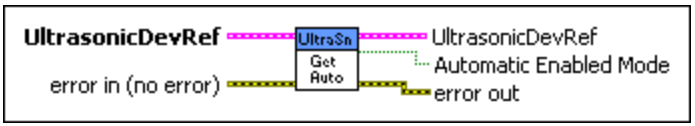
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

GetAutomaticMode.vi

Returns whether the ultrasonic sensor you specify is enabled to take measurements automatically.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function

runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 `4` or `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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

Automatic Enabled Mode returns TRUE when the ultrasonic sensor is enabled to take measurements automatically. Otherwise, **Automatic Enabled Mode** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

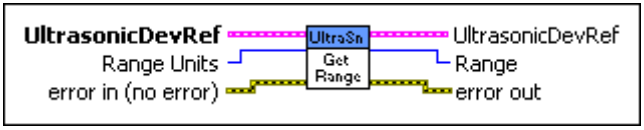
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

GetRange.vi

Returns the distance 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **`4`** or **`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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** returns a value of **`Invalid`**, the VI did not find the counter you specified, and this VI returns an error.

Range 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

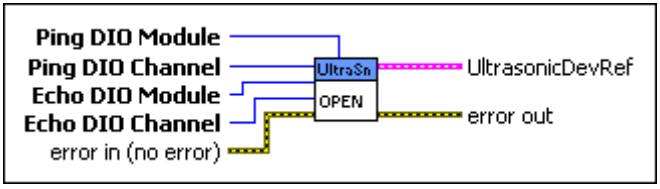
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **4** or **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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

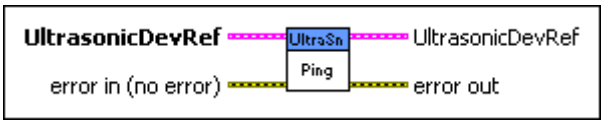
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Ping.vi

Emits a single ping from the ultrasonic sensor you specify. You can use this VI only if the ultrasonic sensor is not enabled to take measurements automatically. Use the GetAutomaticMode VI to determine whether the ultrasonic sensor is enabled to take measurements automatically.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **`4`** or **`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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

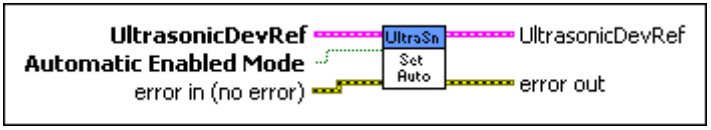
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SetAutomaticMode.vi

Sets whether the ultrasonic sensor you specify is enabled to take measurements automatically.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 7. If **CntIndex** is **Invalid**, this VI returns an error.

Automatic Enabled Mode specifies, when TRUE, that the ultrasonic sensor is enabled to take measurements automatically.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 **`4`** or **`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, unless you specify a different digital module as the default.

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, unless you specify a different digital module as the default.

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 0 and 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

34

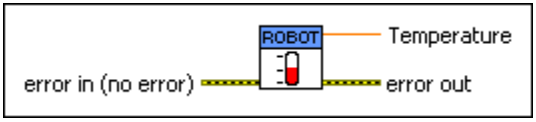
Utilities VIs

Refer to the *LabVIEW Help*, available by selecting **Help»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 ChassisTemperature.vi

Returns the temperature, in degrees Celsius, of the CompactRIO chassis. The CompactRIO device has an operating temperature range of –20 to 55 degrees Celsius.



error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

Temperature returns the temperature, in degrees Celsius, of the CompactRIO chassis.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

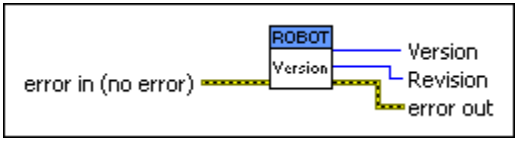
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

FRC FPGAVersion.vi

Returns the version and revision of the FPGA image on the CompactRIO device.



error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

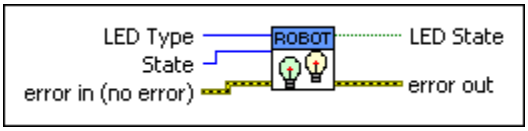
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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. Use the RT LEDs VI to define the USER1 LED. Use the FPGA I/O Node to configure the FPGA LED.



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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select

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

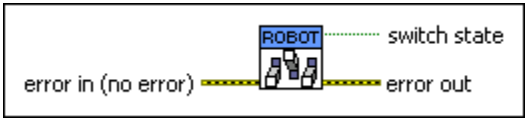
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

35

Victor VIs

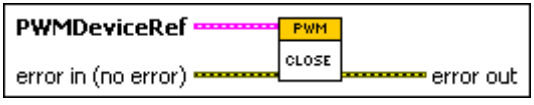
Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Victor VIs.

Use the Victor VIs to set or get the speed of a motor using a Victor motor controller. If you use a Victor motor controller, use the Open VI on this palette instead of the Open VI on the **PWM** palette. The Victor Open VI uses preconfigured PWM range and deadband values specific to the Victor motor controller so you do not need to perform any conversions or scaling calculations on the raw PWM values.

Close.vi

Closes the reference to the PWM, motor controller, or servo you specify. Use this VI to close each PWM reference that you open with the Open VI of the motor controller or servo you use.

The same Close VI appears on the PWM, Jaguar, Victor, and Servo palettes. This Close VI closes references to different motor controller or servo Open VIs. However, you must use the specific motor controller or servo Open VI to open a reference to the device. For example, use the Open VI on the **Jaguar** palette to open a reference to the Jaguar motor controller.



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 PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

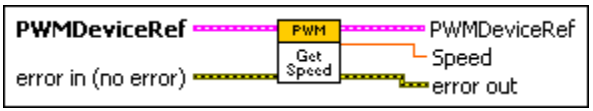
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

GetSpeed.vi

Returns the scaled value from the PWM you specify. The scaled value represents speed. If you use a Jaguar or Victor motor controller, 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.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other Victor VIs or Jaguar VIs, use the Open VI on the **Victor** or **Jaguar** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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. If you use a Jaguar or Victor motor controller, 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

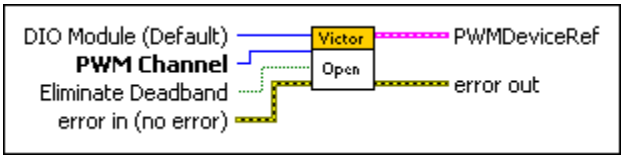
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Open.vi

Creates a reference to the Victor motor controller you specify. You must open a reference before using any other VIs on this palette. After you open a reference to the Victor motor controller, you can use the GetSpeed and SetSpeed VIs to set and get the speed of the motor, respectively.



DIO Module (Default) specifies the slot number on the CompactRIO device of the digital module you want to use. 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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module you want to use. **PWM Channel** can specify a value between 1 and 10. If **PWM Channel** is **Invalid**, this VI returns an error.

Eliminate Deadband, when TRUE, enables a set of scaling constants that eliminate the effects of the deadband in the motor controller. The default is FALSE.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

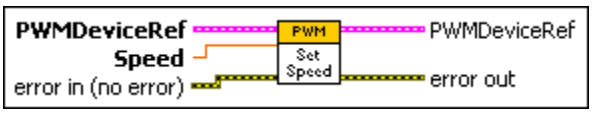
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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.



PWMDeviceRef specifies a reference to the PWM you want to use. When you use this VI with other Victor VIs or Jaguar VIs, use the Open VI on the

Victor or **Jaguar** palette to open this reference. Otherwise, use the general PWM Open 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **4** or **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, unless you specify a different digital module as the default.

PWM Channel specifies the channel of the digital module from which you want to read the PWM value.**PWM Channel** can specify a value between 1 and 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

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

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

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

PWMDeviceRef returns a reference to the PWM.

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.

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

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

code is the error or warning code. Right-click the **error out** indicator on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

source describes the origin of the error or warning. Right-click the **error in** control on the front panel and select **Explain Error** or **Explain Warning** from the shortcut menu for more information about the error.

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 **`4`** or **`6`**. If **DIO Module** returns a value of **`Default`**, the default digital module is used. The default digital module is the first digital module, or the module in slot 4, unless you specify a different digital module as the default.

PWM Channel returns the channel of the **DIO Module** that you want to use. **PWM Channel** can return a value between 1 and 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.

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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

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

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

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

36

Watchdog VIs

Refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**, for the latest information about the Watchdog VIs.

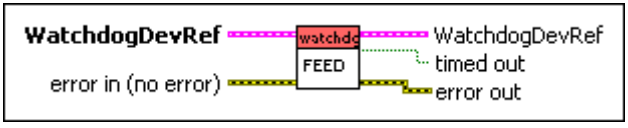
Use the Watchdog VIs to control the user watchdog if you choose to enable the user watchdog.

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 Estop, 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.

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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

timed out returns TRUE if the watchdog was timed out when you fed it. 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

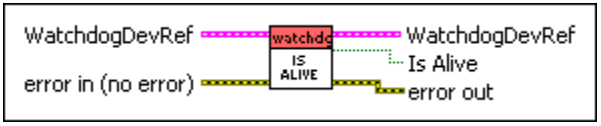
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

IsAlive.vi

Returns whether the user watchdog is alive, or has not timed out. 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Is Alive returns TRUE if the user watchdog is alive. Otherwise, **Is Alive** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

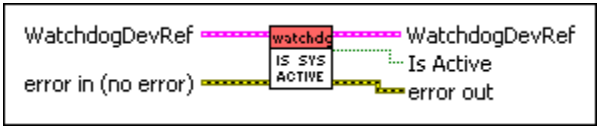
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

IsSystemActive.vi

Returns whether 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, the system is inactive, and the PWM, relay, and solenoid outputs of the CompactRIO device are disabled.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

Is Active 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, **Is Active** 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

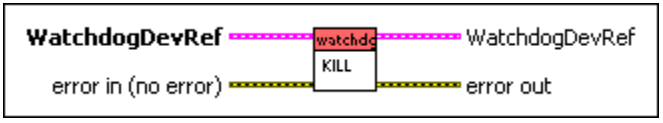
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

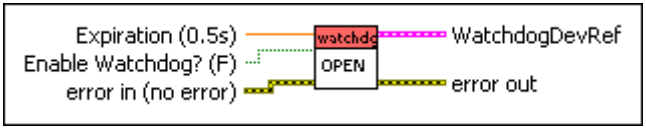
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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.

Enable Watchdog? (F) specifies, when TRUE, that the user watchdog is enabled. The default is FALSE. 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, the watchdog disables the PWM, relay, and solenoid outputs of the CompactRIO device.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

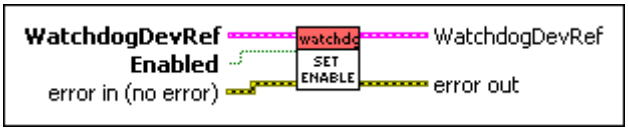
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 Open VI or the SetExpiration VI, the watchdog disables the PWM, relay, and solenoid outputs of the CompactRIO device.

error in (no error) describes the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function

runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

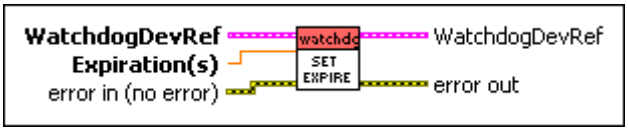
status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.

SetExpiration.vi

Specifies the time 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.



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 VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning. The default is an empty string.

Expiration(s) 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 the error status before this VI or function runs. The default is no error. If an error occurred before this VI or function runs, the VI or function passes the **error in** value to **error out**. This VI or function runs normally only if no error occurred before this VI or function runs. If an error occurs while this VI or function 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 **error in** and **error out** to check errors and to specify execution order by wiring error out from one node to error in of the next node.

status is TRUE (X) if an error occurred before this VI or function ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI or function ran. The default is FALSE.

code is the error or warning code. The default is 0. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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 or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function 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** indicator on the front panel and select **Explain Error** from the shortcut menu for more information about the error.

status is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

code is the error or warning code. If **status** is TRUE, **code** is a nonzero error code. If **status** is FALSE, **code** is 0 or a warning code.

source describes the origin of the error or warning and is, in most cases, the name of the VI or function that produced the error or warning.