# LabVIEW Robotics Programming Guide for *FIRST* Robotics Competition

Use the following tutorials to set up hardware and software for *FIRST* Robotics Competition (FRC) and to develop LabVIEW programs that control the robot.

---

## What's New

The following list describes new and changed features in LabVIEW for FRC 2013:

- **LabVIEW for FRC software—**runs on LabVIEW 2012.
- **Robot Simulation—**allows you to program and run robot code without a CompactRIO FRC device. When you run in simulated mode, the FRC Simulation Viewer opens. The viewer is a simulated environment where you can test drive your simulated robot. The code from a simulated robot can be built for your real robot. For more information see Tutorial 10—Robot Simulation
-

  **Updated Driver Station**
  - **Joystick LED—**lets you know if joysticks are not plugged in.
  - **Test Mode—**allows you to run custom test code.
  - **Driver Station Log Viewer—**improved to log more data. Also can now be run from the Start menu.
  - **Dashboard Type—**lets you choose the programming language of your dashboard. The default is LabVIEW.
- **Updated Dashboard**
  - **Drive and Motors—**displays joystick and motor data.
  - **Test Tab—**shows data from custom tests set up on the robot.
- **WPI Robotics Library VIs**
  - **New Dashboard palette -** includes VIs to easily read and write data between your robot and the dashboard.
- **New Tutorials**
  - Tutorial 10—Robot Simulation
- **New Examples—**include a new I2C Color example, and a new Vision example. Access FRC examples from the Support page of the Getting Started window or by selecting **Help»Find Examples**.

Table of Contents

---

## FRC Terminology

The following terms are useful to know as you build and program a robot with LabVIEW for FRC and as you read this guide.

**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. The CompactRIO device in the FRC kit 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 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 a system watchdog. The system watchdog shuts down all motors if the communication network fails.

**Driver Station**

The driver station is a program that passes data between the host computer; input devices, such as joysticks and game controllers; and the robot. The driver station can read analog data and can 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.

During development you can connect the driver station PC directly to the CompactRIO device using a crossover Ethernet cable. Or, the driver station can communicate with the CompactRIO device through a wireless access point on the robot. During competition you must connect the driver station PC is directly to the Field Management System (FMS) using an Ethernet cable.

Using an Ethernet connection, connect the PC that runs the driver station to the Ethernet router. If you develop programs on a host computer, you also must connect the host computer to the Ethernet router.

**Autonomous and Teleop Modes**

FRC robots can run in two modes: 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 PC using a USB connection. You can use any joystick, game controller, or other input device that you choose.

Most FRC programs include code for both Autonomous and Teleop modes, and then you can decide which code to run from the FMS. The FRC cRIO Robot Project wizard creates template code to include both Autonomous and Teleop code.

**Enabled and Disabled Status**

The FRC software allows you to enable and disable the program running on the CompactRIO device. For example, you might want the program to continue running, but you do not want any part of the robot to move for safety reasons. Setting the Disabled status 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 you set the status of the robot to Enabled status, the system watchdog re-enables the PWM, relay, and solenoid outputs. Therefore, you can develop your robotics program to move motors and drive the robot when the robot is enabled.

Ensure the program you run on the CompactRIO device handles the Disabled and Enabled statuses appropriately. Refer to Tutorial 4—Developing a Robot Project for more information about the robot projects.

**Init, Execute, and Stop Call Contexts**

The robot can be in one of the following states:

- Autonomous Disabled
- Autonomous Enabled
- Teleop Disabled
- Teleop Enabled

For each of these states, the robot can also be in one of three call contexts: Init, Execute, or Stop. Use the Init call context to tell the robot to perform any initialization tasks, such as opening sensor or I/O references. Use the Execute call context to tell the robot to move, read and write data, or perform some other behavior. Use the Stop call context to tell the robot to perform tasks such as stopping motors and closing references.

**Estop State**

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

Table of Contents

## System Setup

A robot controller system might consist of the following subsystems:

a. **Driver Console**—Establishes communication between the CompactRIO device on the robot and the host computer. This subsystem includes a computer that runs the driver station program; one or more input devices, such as joysticks and game controllers; and a host computer.

   **Note** The host computer can be the same computer that runs the driver station program.

b. **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.
c. **Robot**—Consists of all parts that make up the moving robot. This subsystem includes the CompactRIO device with five I/O modules, analog and pneumatic breakouts, a digital sidecar, and an Axis camera.

## Robot Subsystem

This section describes the CompactRIO and Axis camera components of the robot subsystem and then how you can use LabVIEW to interface with this subsystem.

**CompactRIO Device**

The CompactRIO device consists of the following components:

**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, you deploy an application to run on the CompactRIO device from the **RT CompactRIO Target** directory of the **Project Explorer** window. Refer to Tutorial 4—Developing a Robot Project 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 can be reconfigured for different applications. However, the FPGA in the FRC CompactRIO device is already configured for FRC applications, and you cannot change the configuration. Each I/O module on the CompactRIO device interfaces directly with the FPGA.

**I/O Modules**

The CompactRIO device contains the following I/O modules: NI 9201 analog input, NI 9403 digital input/output, and NI 9472 digital output.

The following table shows which modules correspond to which slots on the CompactRIO device.

| Module | cRIO FRC | cRIO FRC II |
|---|---|---|
| NI 9201 | Slot 1 or 5 | Slots 1 |
| NI 9403 | Slot 2 or 6 | Slots 2 |
| NI 9472 | Slot 3 or 7 | Slots 3 |
| | | |

| | | |
|---|---|---|
| Any Module | Any Module | Slots 4 |

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 9403 in slot 2 of the CompactRIO device, or you specify a module value of 2 to use the NI 9403 in slot 5. You then can select a value for the channel on the NI 9403 that you want to use.

The following table describes the modules on the CompactRIO device.

| Module | Description |
|---|---|
| NI 9201 | The NI 9201 provides connections for eight analog input channels. You connect an analog breakout to each of the NI 9201 modules. You then can connect any analog sensors, such as accelerometers and gyros, to the analog breakouts to acquire analog data.<br><br>**Note** You must insert the NI 9201 analog modules before the CompactRIO device boots for accurate calibration of the modules. Swapping the modules after the CompactRIO device boots applies the calibration for the original module to the replacement module. Inserting the modules after the CompactRIO device boots applies factory default calibration values as the analog module constants.<br><br>Use the Accelerometer VIs and the Gyro VIs to acquire analog data from those sensors. Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use these VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.<br><br>Use the AnalogChannel VIs to acquire analog data from other analog input devices. Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use these VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.<br><br>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.<br><br>Use the Counter, Encoder, and Ultrasonic VIs to acquire digital data from those sensors. Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use these VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.<br><br>Use the DigitalInput VIs and the DigitalOutput VIs to send or receive digital data from other digital input and output devices. Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use these VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.<br><br>Refer to the *NI 9403 Operating Instructions and Specifications* document, accessible by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `NI_9403_Operating_Instructions.pdf`, for more information about the NI 9403. |
| NI 9472 | The NI 9472 provides connections for eight digital output channels. Whereas the NI 9403 can output a maximum voltage of 5.2 V, the NI 9472 can output a maximum voltage of 30 V. If you use the pneumatic breakout with the NI 9472, the NI 9472 can output a maximum voltage of 12 V.<br><br>You can use this module to control solenoids. Connect a pneumatic breakout to the NI 9472 and connect solenoids to the pneumatic breakout. Then use the Solenoid VIs to specify whether the NI 9472 sends a 12 V signal to a solenoid.<br><br>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.

Click the **Find FRC Examples** link on the **Getting Started** window to find example programs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.

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.

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.

**Note** You can send image data to the host computer or perform live front panel debugging only during development.

Before you can use the Axis camera, you must configure how to connect the camera. Use the Setup Axis Camera Tool, available by selecting **Tools»Setup Axis Camera**, to configure the Axis camera.

**LabVIEW for FRC**

LabVIEW for FRC includes LabVIEW 2011, the LabVIEW Real-Time Module, the NI Vision Development Module, the NI-RIO driver, and 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 LabVIEW to develop a program that runs on the host computer or that you can deploy and run on the CompactRIO device.

- **FRC CompactRIO VIs**—The FRC CompactRIO 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.

  Click the **Find FRC Examples** link on the **Getting Started** window to find example programs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.

- **Setup Axis Camera Tool**—Use the Setup Axis Camera Tool, available by selecting **Tools»Setup Axis Camera**, to choose to connect the Axis camera using the Robot Radio or the Ethernet port 2 on the 8-slot CompactRIO device.

  Refer to Tutorial 2—Setting up the Axis Camera 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 also can restore an image on the CompactRIO device or update the CompactRIO device with a new name or IP address.

Refer to Tutorial 1—Setting up the CompactRIO for more information about configuring the CompactRIO device.

Table of Contents

## Tutorial 1—Setting up the CompactRIO Device

You must configure the CompactRIO device and the Axis camera before you can run a robotics program on the robot.

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

### A) Setting the Static IP Address of the Computer

Windows 7
Windows XP

**(Windows 7)** Complete the following steps to set the static IP address of a computer running Windows 7.

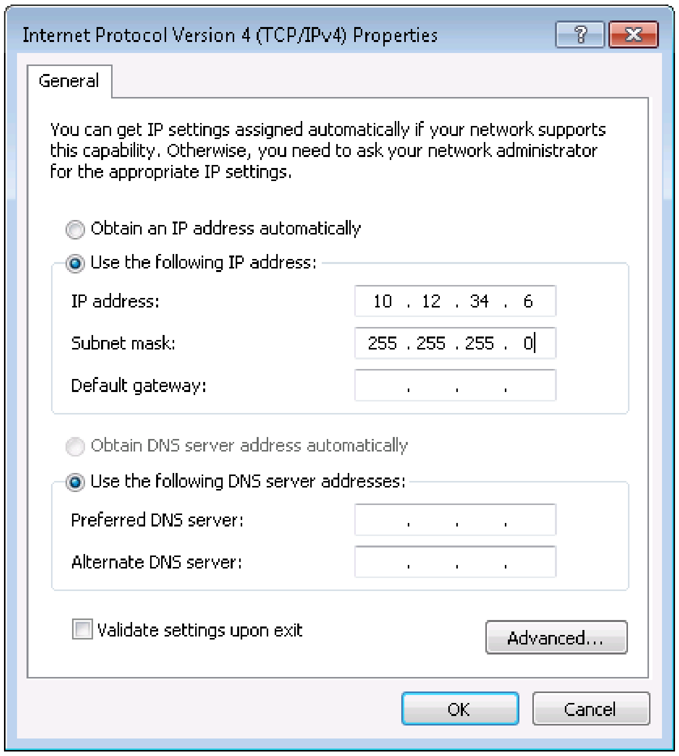
**Note** If the computer has more than one network connection, you must disable all network connections except for the connection you will use to connect to the CompactRIO Device.

1. Select **Start»Control Panel** and then search for the phrase **View network connections** in the search field.
2. Click the **View network connections** link.
3. Double-click **Local Area Connection** to display the **Local Area Connection Status** dialog box.
4. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
5. Double-click **Internet Protocol Version 4 (TCP/IPv4)** from the **This connection uses the following items** list to display the **Internet Protocol Version 4 (TCP/IP) Properties** dialog box.
6. Select the **Use the following IP address** option.
7. In the **IP address** text box, enter `10.xx.yy.6`, where `yy` corresponds to the last two digits of the team number and `xx` corresponds to the first or first two digits of the team number.

The following table lists examples of the static IP addresses that correspond to different team numbers.

| Team Number | Static IP Address |
|---|---|
| 64 | 10.0.64.6 |
| 512 | 10.5.12.6 |
| 1234 | 10.12.34.6 |

8. In the **Subnet mask** text box, enter `255.255.255.0`.



You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically
[✓ Use the following IP address:]
IP address: 10 . 12 . 34 . 6
Subnet mask: 255 . 255 . 255 . 0
Default gateway:

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]
Preferred DNS server:
Alternate DNS server:

Validate settings upon exit

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

**(Windows XP)** Complete the following steps to set the static IP address of a computer running Windows XP.

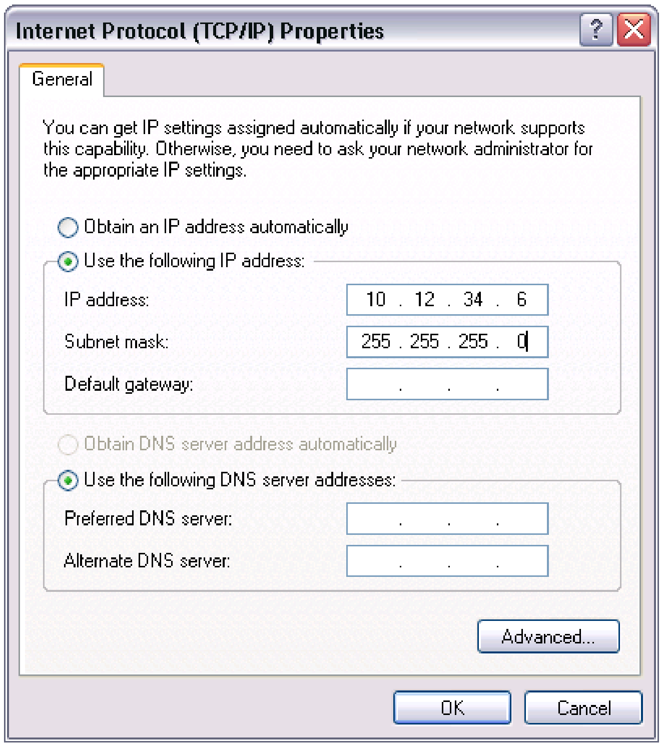
**Note** If the computer has more than one network connection, you must disable all network connections except for the connection you will use to connect to the CompactRIO Device.

1. Select **Start»Control Panel»Network Connections»Local Area Connection** to display the **Local Area Connection Status** dialog box.
2. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
3. On the **General** page, select **Internet Protocol (TCP/IP)** from the **This connection uses the following items** list.
4. Click the **Properties** button to display the **Internet Protocol (TCP/IP) Properties** dialog box.
5. Select the **Use the following IP address** option.
6. In the **IP address** text box, enter `10.xx.yy.6`, where `yy` corresponds to the last two digits of the team number and `xx` corresponds to the first or first two digits of the team number.

The following table lists examples of the static IP addresses that correspond to different team numbers.

| Team Number | Static IP Address |
|---|---|
| 64 | 10.0.64.6 |
| 512 | 10.5.12.6 |
| 1234 | 10.12.34.6 |

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


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:
IP address: 10 . 12 . 34 . 6
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

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.

**B) Connecting the Computer to the CompactRIO Device**

After you set the static IP address of the host 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.

**C) Running the CompactRIO Imaging Tool**

**Considerations Before Running the CompactRIO Imaging Tool**

- Before configuring the CompactRIO device with the CompactRIO Imaging Tool, ensure that the hardware and software are configured properly by completing section A, *Setting the Static IP Address of the Computer* above.
- Do not use the CompactRIO Imaging Tool on the CompactRIO device over a wireless connection. If the connection is lost, the data that the CompactRIO Imaging Tool writes to the CompactRIO device is corrupted.
- Do not use Measurement & Automation Explorer (MAX) to install additional software on the CompactRIO device. MAX overwrites the FRC VIs on the CompactRIO device. If you use MAX to install additional software on the CompactRIO device, you must use the CompactRIO Imaging Tool to restore the device to a usable state.
- Before running the CompactRIO Imaging Tool, ensure the SAFE MODE switch on the CompactRIO device is turned off. For routine use, do not use the CompactRIO Imaging Tool when the CompactRIO device is in SAFE MODE.
- Severe corruptions of the software or settings on the CompactRIO device result in the device no longer functioning. If the CompactRIO device is corrupted or if the IP Address is set incorrectly, the device boots only in SAFE MODE. When this occurs, switch the device into SAFE MODE. The CompactRIO Imaging Tool offers to reformat the disk. After the disk has been reformatted, switch the CompactRIO device out of SAFE MODE, reboot, and run the CompactRIO Imaging Tool normally.

**Running the CompactRIO Imaging Tool**

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

1. Select **Start»All Programs»National Instruments»LabVIEW 2011»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. Place a checkmark in the **Format Controller** checkbox. Use the **Format Controller** section to restore an image on the CompactRIO device or

update the CompactRIO device with a new name or team ID.

4. From the **Select Image** list, select the most recent `.zip` file to download the most recent image to the CompactRIO device. If you do not see a `.zip` file, you must install the FRC LabVIEW Update Suite.
5. Enter the name you want to use to identify the CompactRIO device in the **Device name** text box.
6. Enter your team number in the **Team ID** text box. The CompactRIO Imaging Tool sets the IP address of the CompactRIO device to 10.xx.yy.2, where yy corresponds to the last two digits of the team number and xx corresponds to the first or first two digits of the team number.
7. Click the **Apply** button to apply the changes you made and download the 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.

**Note** If the cRIO Imaging Tool fails to update the CompactRIO device, navigate to the Advanced page of the Internet Protocol (TCP/IP) Properties dialog box, and remove any IP addresses other than your team IP address. Then run the tool again.

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

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

You also can determine the processor load of the CompactRIO device by using the command `memShow` in the Net Console for FRC cRIO Utility.

Table of Contents

## Tutorial 2—Setting up the Axis Camera

You must configure the CompactRIO device and the Axis camera before you can run a robotics program on the robot.

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.

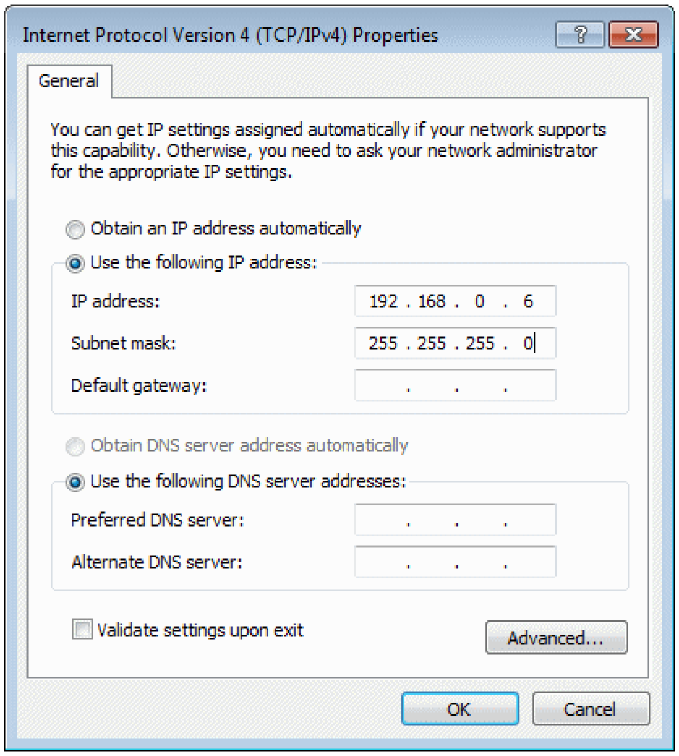
To configure the Axis camera, you must A) set the static IP address of the computer, B) connect the computer to the camera, and then C) run the Setup Axis Camera Tool.

**A) Setting the Static IP Address of the Computer**

Windows 7
Windows XP

**(Windows 7)** Complete the following steps to set the static IP address of a computer running Windows 7.

1. Select **Start»Control Panel** and then search for the phrase **View network connections** in the **search** field.
2. Click the **View network connections** link.
3. Double-click **Local Area Connection** to display the **Local Area Connection Status** dialog box.
4. Click the **Properties** button to display the **Local Area Connection Properties** dialog box.
5. Double-click **Internet Protocol Version 4 (TCP/IPv4)** from the **This connection uses the following items** list to display the **Internet Protocol Version 4 (TCP/IP) Properties** dialog box.
6. Select the **Use the following IP address** option.
7. In the **IP address** text box, enter `192.168.0.06`.
8. The **Subnet mask** text box defaults to 255.255.255.0. Use this default value.


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:
IP address: 192 . 168 . 0 . 6
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

○ Obtain DNS server address automatically
● Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

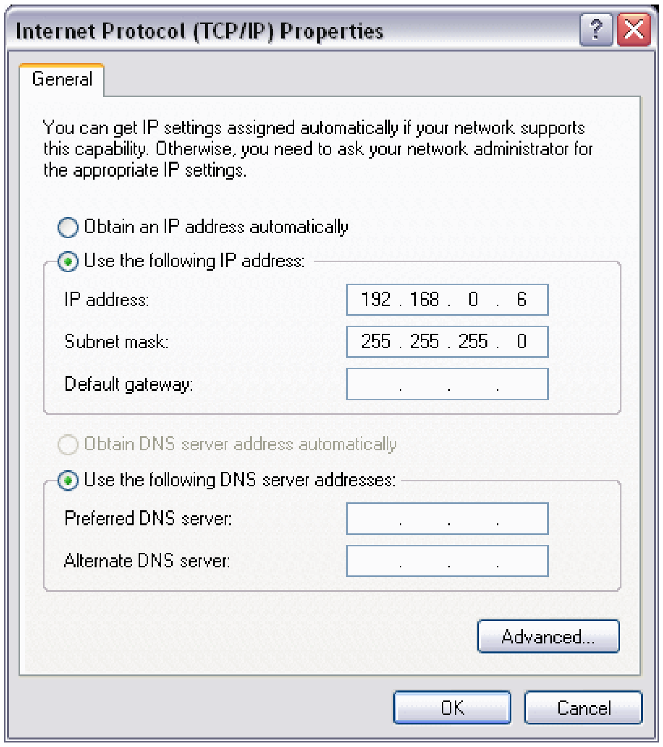
☐ Validate settings upon exit

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

10. Click the **Close** button to close the **Local Area Connection Status** dialog box.

**(Windows XP)** Complete the following steps to set the static IP address of a computer running Windows XP.

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 `192.168.0.6`.
7. In the **Subnet mask** text box, enter `255.255.255.0`.


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

( ) Obtain an IP address automatically
(●) Use the following IP address:
IP address: 192 . 168 . 0 . 6
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

( ) Obtain DNS server address automatically
(●) Use the following DNS server addresses:
Preferred DNS server: . . .
Alternate DNS server: . . .

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.

**B) Connecting the Computer to the Axis Camera**

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.

Ensure that the camera has 5V DC power, which is typically connected to the 5V supply on the Power Distribution Board.

**C) 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 2011»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. Select how you want to connect the camera. The **Setup Axis Camera** dialog box allows you to choose to connect through the Robot Radio or through Ethernet port 2 on an 8-slot CompactRIO device. If you select the Robot Radio option, enter your Team ID.
3. Click **Apply** to enter your selections
4. Click the **Close** button to close the **Setup Axis Camera** dialog box.

If you choose to connect through Ethernet port 2 of the 8-slot CompactRIO device, use an Ethernet crossover cable to connect the camera to the device. The program you run on the CompactRIO device then can communicate with the camera.

Table of Contents

## Tutorial 3—Setting up the Robot Radio

Complete the steps in the following sections to configure the D-Link DAP-1522 Radio.

**Resetting the Radio to Its Default Setting**

1. Set the D-Link unit to **Bridge** mode by changing the switch on the back of the unit.
2. Plug in the power and Ethernet connections.
3. Wait for the orange bridge light to flash.
4. Hold the **reset** button on the back of the unit for 10 seconds.
5. Wait for the orange bridge light to stop flashing, which indicates the radio is resetting.
6. Wait for the orange bridge light to resume flashing; the radio is now reset.

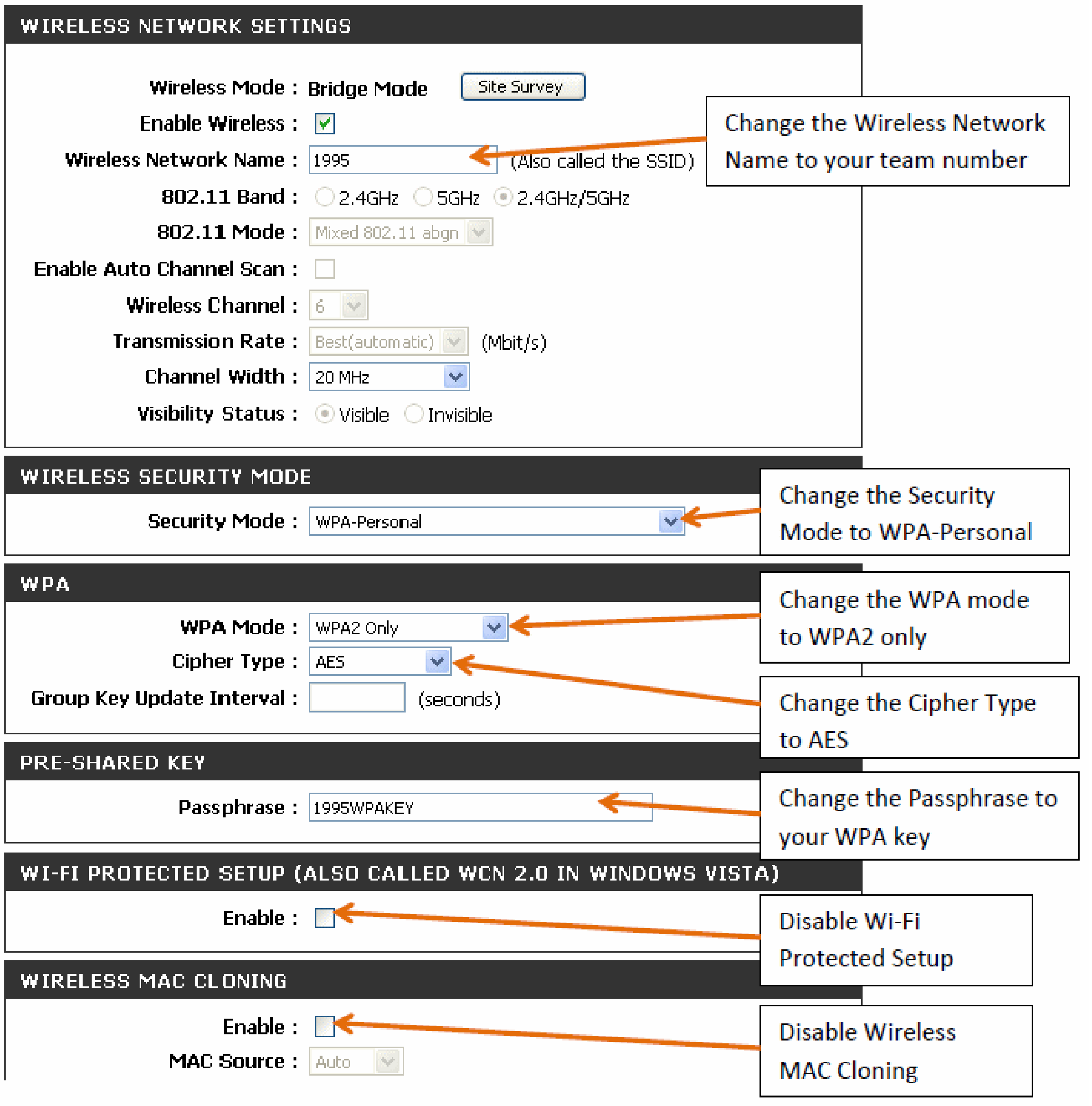
**Note** The figures in this tutorial reflect an example radio configured for team 1995. You must use your team number when configuring your radio.

**Configuring the Radio for Competitions**

Complete the following steps to configure a radio for competition.

**Note** At official FRC events, you must use the FRC Radio Kiosk at the inspection table to configure your radio. You can download a team-use version of the radio configuration kiosk on the FIRST kit of parts Web page.

1. Set the computer IP address to 192.168.0.51.
2. Set the D-Link unit to **Bridge** mode by changing the switch on the back of the unit. The radio must be in Bridge mode at competitions.
3. Reset the radio, as described in the previous section.
4. Connect the radio to the computer using an Ethernet cable.
5. Launch Internet Explorer and type in the address 192.168.0.50 in the address bar.
    - Username = admin
    - Password = blank
6. Click the **Setup** tab on the top menu bar.
7. Click the **Wireless** tab on the left side of the page.
8. Change the **Wireless Network Settings** to match the settings shown in the following figure.



WIRELESS NETWORK SETTINGS

Wireless Mode : Bridge Mode
Enable Wireless : ✓
Wireless Network Name : 1995 (Also called the SSID)
802.11 Band : 2.4GHz 5GHz [✓ 2.4GHz/5GHz]
802.11 Mode : Mixed 802.11 abgn
Enable Auto Channel Scan : ☐
Wireless Channel : 6
Transmission Rate : Best(automatic) (Mbit/s)
Channel Width : 20 MHz
Visibility Status : [✓ Visible] Invisible

WIRELESS SECURITY MODE

Security Mode : WPA-Personal

WPA

WPA Mode : WPA2 Only
Cipher Type : AES
Group Key Update Interval : (seconds)

PRE-SHARED KEY

Passphrase : 1995WPAKEY

WI-FI PROTECTED SETUP (ALSO CALLED WCN 2.0 IN WINDOWS VISTA)

Enable : ☐

WIRELESS MAC CLONING

Enable : ☐
MAC Source : Auto

Change the Wireless Network Name to your team number

Change the Security Mode to WPA-Personal

Change the WPA mode to WPA2 only

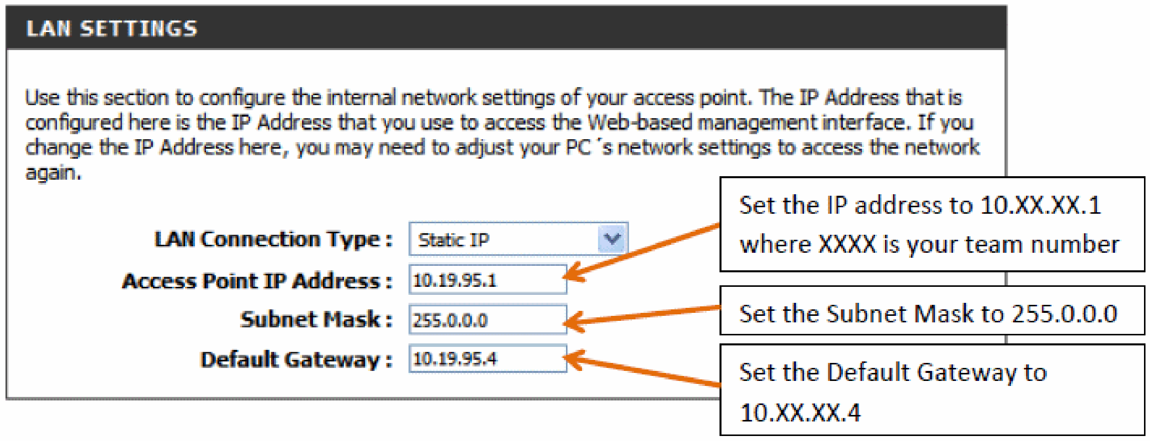
Change the Cipher Type to AES

Change the Passphrase to your WPA key

Disable Wi-Fi Protected Setup

Disable Wireless MAC Cloning

9. Save the settings.
10. Click the **Network Settings** tab on the left side of the page.
11. Change the network settings to match the settings shown in the following figure.



**LAN SETTINGS**

Use this section to configure the internal network settings of your access point. The IP Address that is configured here is the IP Address that you use to access the Web-based management interface. If you change the IP Address here, you may need to adjust your PC's network settings to access the network again.

| | |
|---|---|
| LAN Connection Type : | Static IP |
| Access Point IP Address : | 10.19.95.1 |
| Subnet Mask : | 255.0.0.0 |
| Default Gateway : | 10.19.95.4 |

Set the IP address to 10.XX.XX.1 where XXXX is your team number

Set the Subnet Mask to 255.0.0.0

Set the Default Gateway to 10.XX.XX.4

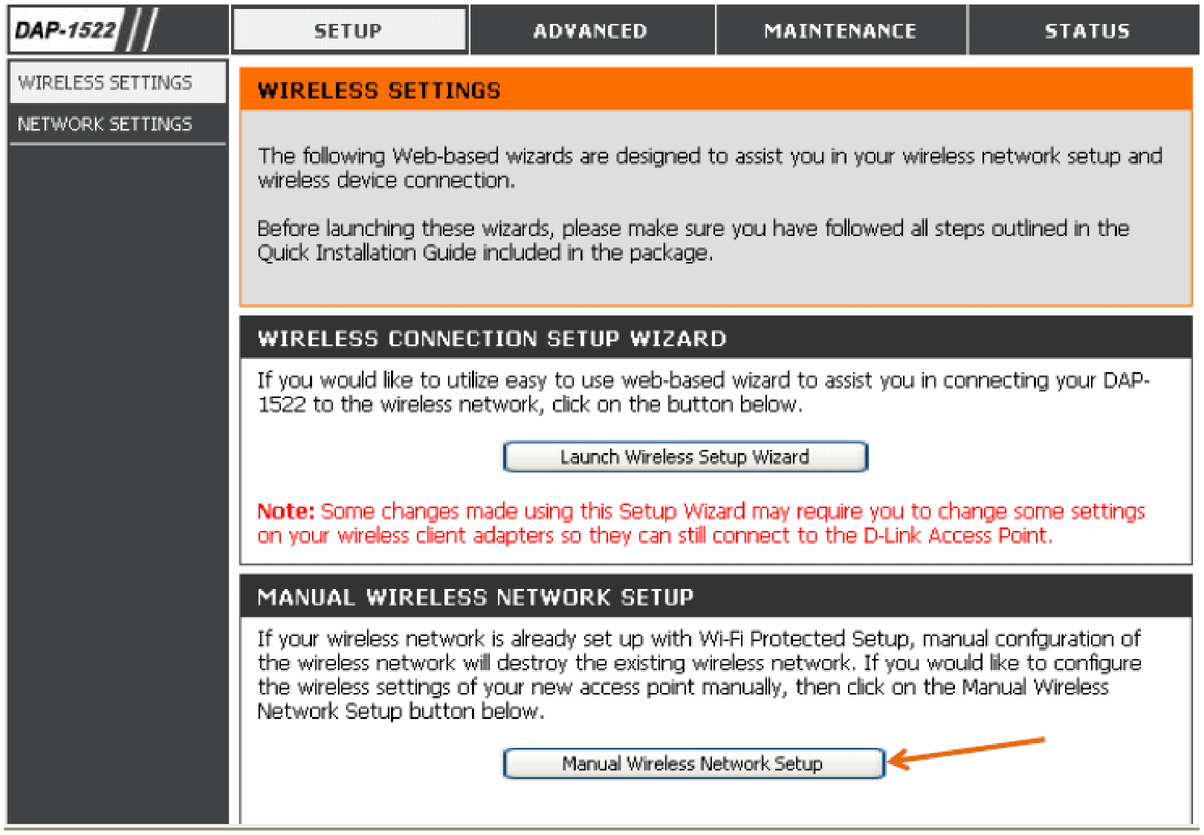
12. Save the settings.

**Configuring the Radio to Use an Access Point**

Complete the following steps to configure the radio to use an access point.

**Note** If the computer connects to the radio wirelessly, complete the following steps after you configure the radio for Bridge mode.

1. Set the D-Link unit to **AP** mode by changing the switch on the back of the unit.
2. Ensure the computer is set to the proper IP address (192.168.0.51 if you have not changed the radio network settings or 10.XX.XX.51 if you have previously configured the radio).
3. Connect the radio to the computer using an Ethernet cable.
4. Launch Internet Explorer and type in the radio IP address, which is 192.168.0.50 or 10.XX.XX.1, depending on whether you previously changed the radio network.
    - Username = admin
    - Password = blank
5. Click the **Manual Wireless Network Setup** button, shown as follows.



**WIRELESS SETTINGS**

The following Web-based wizards are designed to assist you in your wireless network setup and wireless device connection.

Before launching these wizards, please make sure you have followed all steps outlined in the Quick Installation Guide included in the package.

**WIRELESS CONNECTION SETUP WIZARD**

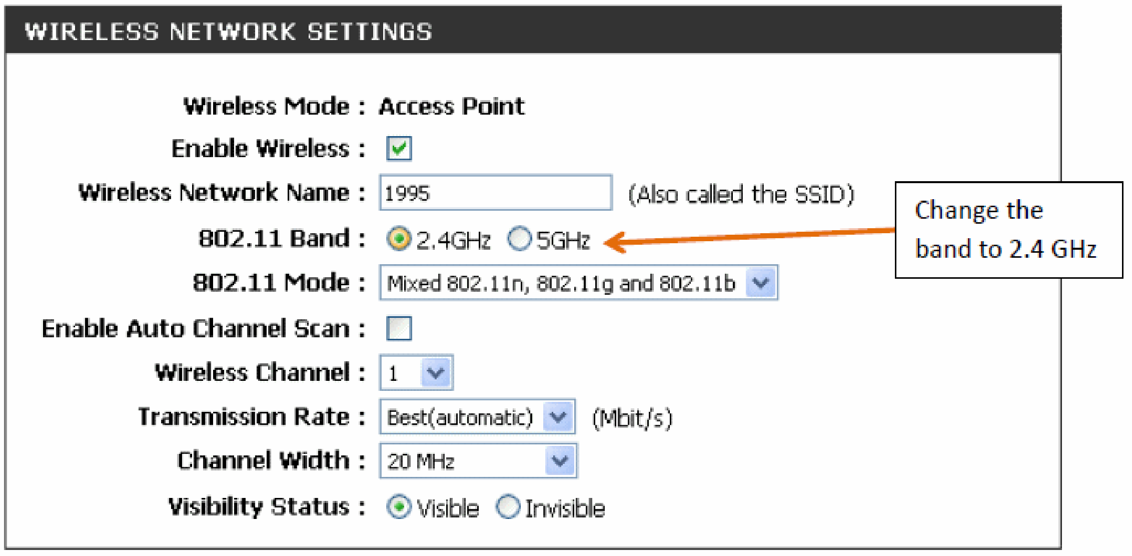
If you would like to utilize easy to use web-based wizard to assist you in connecting your DAP-1522 to the wireless network, click on the button below.

**Note:** Some changes made using this Setup Wizard may require you to change some settings on your wireless client adapters so they can still connect to the D-Link Access Point.

**MANUAL WIRELESS NETWORK SETUP**

If your wireless network is already set up with Wi-Fi Protected Setup, manual confguration of the wireless network will destroy the existing wireless network. If you would like to configure the wireless settings of your new access point manually, then click on the Manual Wireless Network Setup button below.

6. Change the 802.11 Band to 2.4Ghz, shown as follows.


WIRELESS NETWORK SETTINGS

Wireless Mode : Access Point

Enable Wireless : ✓

Wireless Network Name : 1995 (Also called the SSID)

802.11 Band : [✓ 2.4GHz] 5GHz

802.11 Mode : Mixed 802.11n, 802.11g and 802.11b

Enable Auto Channel Scan : ☐

Wireless Channel : 1

Transmission Rate : Best(automatic) (Mbit/s)

Channel Width : 20 MHz

Visibility Status : [✓ Visible] Invisible

Table of Contents

---

## Tutorial 4—Developing a Robot Project

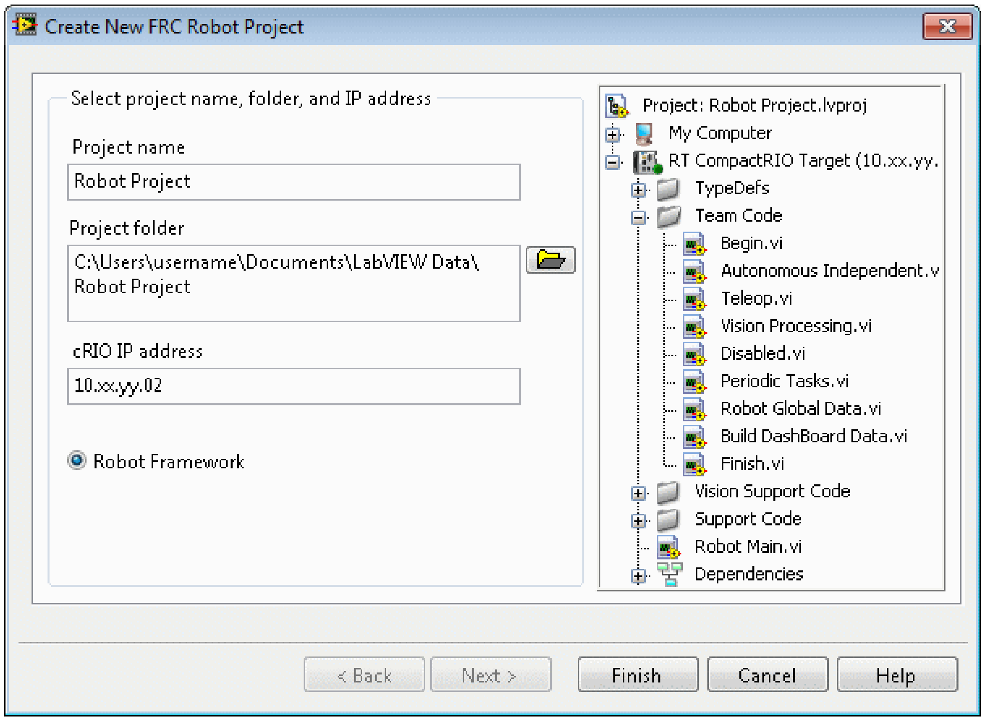
The *FIRST* Robotics Competition (FRC) projects are sets of VIs, organized in LabVIEW projects, that you can use as a template when building a robotics application. The FRC projects consist of two project templates. Use the FRC robot project template to develop the program you want to deploy to and run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

This tutorial describes how to create, modify, and deploy a robot project, also referred to as an FRC robot project, to the CompactRIO device. In this tutorial, you develop a program to perform tank driving with two joysticks and Jaguar motor controllers.

### Creating an FRC Robot Project

Complete the following steps to create an FRC robot project.

1. Click the **FRC cRIO Robot Project** link in the **Getting Started** window to display the **Create New FRC Robot Project** dialog box, shown as follows.


Create New FRC Robot Project

Select project name, folder, and IP address

Project name: Robot Project

Project folder: C:\Users\username\Documents\LabVIEW Data\Robot Project

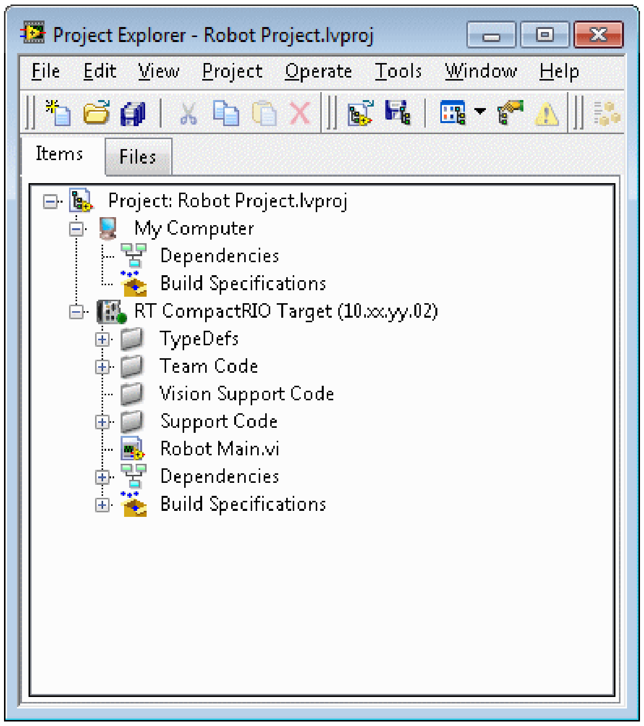
cRIO IP address: 10.xx.yy.02

[✓ Robot Framework]

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

   You can use the CompactRIO Imaging Tool to set the IP address of the CompactRIO device. Refer to Tutorial 1—Setting up the CompactRIO for more information about configuring the CompactRIO device.
5. Click the **Finish** button to close the **Create New FRC Robot Project** dialog box and create the new FRC robot project.

LabVIEW displays the new FRC robot project in the **Project Explorer** window, shown as follows.


Project Explorer - Robot Project.lvproj
File Edit View Project Operate Tools Window Help
Items
Files
Project: Robot Project.lvproj
My Computer
Dependencies
Build Specifications
RT CompactRIO Target (10.xx.yy.02)
TypeDefs
Team Code
Vision Support Code
Support Code
Robot Main.vi
Dependencies
Build Specifications

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

The **RT CompactRIO Target** contains one top-level VI, the Robot Main VI, which is the master VI for the robot and is the top-level VI for the robotics program you run on the CompactRIO device. This target also contains a **TypeDefs** folder and a **Team Code** folder. These folders contain type definitions and VIs that the Robot Main VI calls.

**Note** The project that you create also contains default code to drive a robot in Teleop mode. The default code assumes the robot includes two motors and one joystick.

Refer to Tutorial 5—Editing Team Code VIs for more information about the type definitions and subVIS the FRC Robot Project contains.

### Deploying the FRC Robot Project to the CompactRIO

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

#### Deploying the Program Using the Run Button

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.

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

1. In the **Project Explorer** window, double-click the **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 Robot Main VI and any support files for the VI 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 Jaguar motor controllers controlling the two wheels, you can move the joysticks and observe how the robot responds.
3. Move the joysticks and observe how the robot responds.
4. Click the **Abort** button of the Robot Main VI. Notice that the VI stops.

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

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

#### Deploying the Program from the Project Explorer Window

In the **Project Explorer** window, right-click the **Robot Main VI** and select **Deploy** from the shortcut menu to deploy the VI and any support files for the VI to the target. VIs, libraries, and shared variables are downloaded to memory on the CompactRIO device. When you deploy a program from the **Project Explorer** window, the program runs only on the CompactRIO device to which it was deployed. Therefore, you cannot perform live front panel debugging.

#### Building and Deploying a Stand-Alone Application

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

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

1. In the **Project Explorer** window, double-click the **FRC Robot Boot-up Deployment** build specification under the **Build Specifications** folder to display the **FRC Robot Boot-up Deployment 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** text box.
5. Specify the location on the CompactRIO device to which you want to save the stand-alone application in the **Target destination directory**

text box.

6. Select **Source Files** from the **Category** list.
7. Verify that the Robot Main VI is in the **Startup VIs** list.
8. Click the **OK** button to close the **FRC Robot Boot-up Deployment Properties** dialog box.
9. In the **Project Explorer** window, right-click the build specification and select **Build** from the shortcut menu to build the application.
10. After the build finishes, right-click the build specification and select **Run as startup** from the shortcut menu to set the application as the startup application and deploy the application to the CompactRIO device. LabVIEW prompts you to reboot the RT target.
11. Reboot the CompactRIO device to run the application.

    When the CompactRIO device reboots, the status light on the CompactRIO momentarily turns on and then off. If you do not see the status light turn on, you can reboot the CompactRIO device from the **Diagnostic** page of the driver station.
12. Ensure the driver station is running and set to Teleop Enabled mode.
13. Move the two joysticks and observe how the motors of the robot respond.
14. If you no longer want the application to run on the CompactRIO device at startup, right-click the build specification and select **Unset** as startup from the shortcut menu.

**Connecting to the CompactRIO Device**

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. 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.

Table of Contents

## Tutorial 5—Editing Team Code VIs

Use the following VI descriptions to learn which VIs you want to edit to write the program you want to run on the robot.

The **RT CompactRIO Target** contains one top-level VI, the Robot Main VI, which is the master VI for the robot and is the top-level VI for the robotics program you run on the CompactRIO device. This target also contains a **TypeDefs** folder and a **Team Code** folder. These folders contain type definitions and VIs that the Robot Main VI calls.

**Robot Main VI**

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

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

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

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

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

You can modify the code within each of the VIs in the **Team Code** folder of the robot project.

**Team Code VIs**

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

**Begin VI**

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

By default, the Begin VI opens a reference to the Axis camera, opens a reference for two motors, and opens a reference for one joystick. You can use the WPI Robotics Library VIs and other LabVIEW VIs to configure other initialization tasks.

Create all references you want to reuse in the Begin VI before you develop the programs you want to run on the robot. Complete the following steps to create a reference.

1. In the **Project Explorer** window of the FRC robot project, within the **Team Code** folder under the **RT CompactRIO Target**, double-click the **Begin.vi** item to open the Begin VI.
2. Select **Window»Show Block Diagram** or press the <Ctrl–E> keys to view the block diagram.
3. Notice that the code already contains references for Camera, Left and Right Motors, and Joystick 1.

   Similarly, you can add references for any other sensor or actuator. Refer to Tutorial 7—Integrating Examples into Robot Code for information about adding a reference for a gyroscope.

After you set references in the Begin VI, use the RefNum Registry Get VIs to reuse the references in other VIs that you create.

**Autonomous Independent VI**

The Autonomous Independent VI runs while the robot is in the Autonomous mode. You do not need to program the Autonomous Independent VI to stop after a certain time because the Robot Main VI terminates this VI when the mode changes to Teleop.

By default, the block diagram of the Autonomous Independent VI contains a Diagram Disable structure. The Diagram Disable structure contains default autonomous code for the robot. By default, this code is disabled. You can enable this code by right-clicking the edge of the structure and selecting **Remove Diagram Disable Structure** or **Enable This Subdiagram**. You also can delete the default autonomous code and use the WPI Robotics Library VIs or other LabVIEW VIs to specify the program you want to run while the robot is in the Autonomous mode.

**Disabled VI**

The Disabled VI runs each time the program receives a disabled driver station packet. You can use the Disabled VI to calibrate sensors or the Axis camera.

**Teleop VI**

The Teleop VI iterates and performs some action each time a packet specifying the Teleop mode arrives from the driver station.

The block diagram of the Teleop VI contains basic code to read values from a joystick and update the robot motors. Edit this code or add your own code to drive the robot. If you want to add initialization or clean up code, use call context to control a Case structure with Init, Execute, and Stop cases.

The Teleop VI uses one or more of the references you set in the Begin VI to specify device references of the correct data types for use with the WPI Robotics Library VIs. Refer to the Begin VI section for information about creating references.

**Robot Global Data VI**

Use the Robot Global Data VI to access and pass data among several VIs.

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

On the block diagram of the Vision Processing VI, the **Enable Vision** global variable is wired to the Case structure that determines whether to start or stop image acquisition. The value of the **Enable Vision** control in the Robot Global Data VI determines which case of the Case structure in the Vision Processing VI to execute. For example, you can set the **Enable Vision** global variable to True or False, depending on a joystick button that is read in the Teleop VI. Using the variable this way allows you to turn vision on or off by pressing a joystick button.

**Vision Processing VI**

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

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

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

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

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

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

**Periodic Tasks VI**

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

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

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

The Periodic Tasks VI uses one or more of the references you set in the Begin VI to specify device references of the correct data types for use with the WPI Robotics Library VIs. Refer to the Begin VI section for information about creating references.

**Build DashBoard Data VI**

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

**Finish VI**

The Finish VI performs cleanup tasks before the Robot Main VI stops. This VI runs when you press the **Finish** button on the front panel of the Robot Main VI. This VI does not get called during competition.

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

The Finish VI uses one or more of the references you set in the Begin VI to specify device references of the correct data types for use with the WPI Robotics Library VIs. Refer to the Begin VI section for information about creating references.

**Type Definitions**

The **TypeDefs** folder of the FRC robot project contains type definitions for specifying data types that you use with the WPI Robotics Library VIs. For example, the Dashboard Datatype type definition specifies data that corresponds to the analog module, digital module, and solenoid module of the CompactRIO device. You can add or remove the controls that this type definition contains depending on the data you want to reuse. You also can create your own type definitions to reuse. Refer to Tutorial 7: Creating a Dashboard Project for more information about creating and modifying type definitions.

Table of Contents

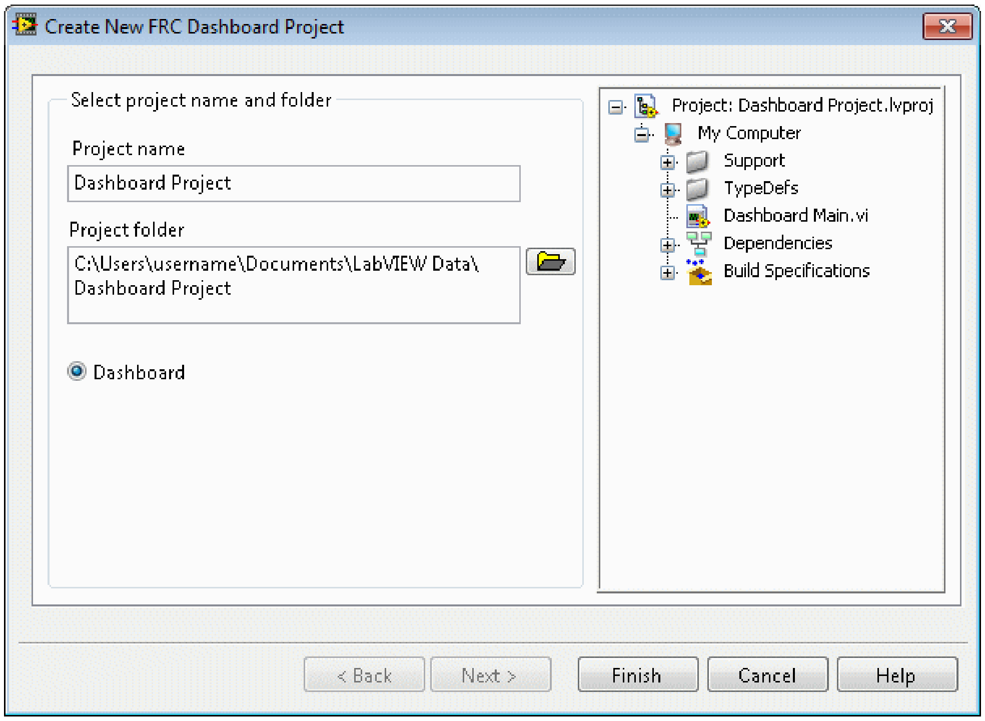
---

## Tutorial 6—Creating a Custom Dashboard

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

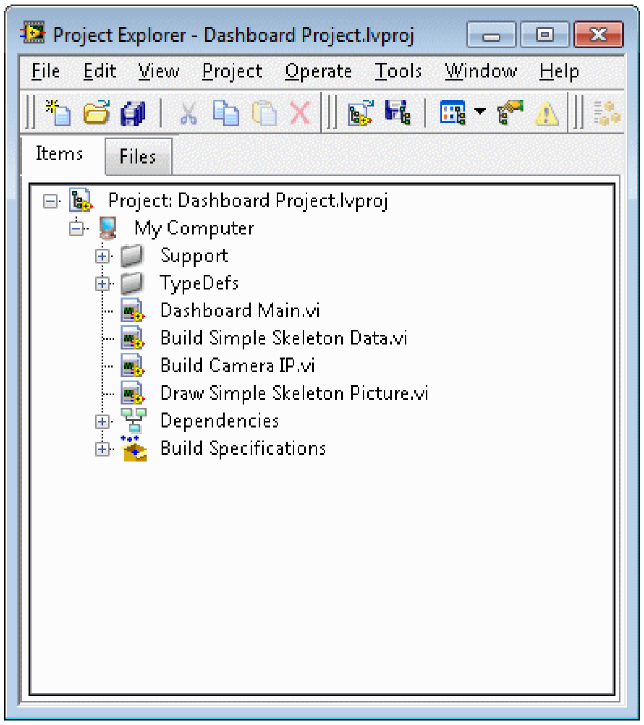
Complete the following steps to create an FRC dashboard project.

1. Click the **FRC Dashboard Project** link in the **Getting Started** window to display the **Create New FRC Dashboard Project** dialog box, shown as follows.



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

The FRC dashboard project looks similar to the following figure:


Project Explorer - Dashboard Project.lvproj
File Edit View Project Operate Tools Window Help
Items
Files
Project: Dashboard Project.lvproj
My Computer
Support
TypeDefs
Dashboard Main.vi
Build Simple Skeleton Data.vi
Build Camera IP.vi
Draw Simple Skeleton Picture.vi
Dependencies
Build Specifications

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

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

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

You also can use the Dashboard Main VI to read information about the robot, such as error information, robot status, and battery level. In the **Project Explorer** window, double-click the **DashBoard Main.vi** item to open the Dashboard Main VI. By default, the front panel displays the following information:

- Image—Displays the latest image that the camera on the CompactRIO device acquired. The **Video Enable Boolean** control in this section turns image display on or off. Image display might slow performance.
- Kinect Skeleton—Displays the latest image that the Kinect device reads. A green figure represents a player's skeleton with green circles representing individual skeleton joints. White circles represent detected players.
- Analag #1—Displays analog input values from the NI 9201 modules in the CompactRIO device.

- Digital #1—Displays digital input or digital output values from the NI 9403 modules in the CompactRIO device.
- Solenoid #1—Displays digital output values from the NI 9472 module in the CompactRIO device. This module often is used to control a solenoid.
- Communications—Displays input and output values from the driver station.

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

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

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

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

**Modifying the FRC Dashboard Project**

The Dashboard Main VI in the FRC Dashboard Project displays information about the robot, such as error information, robot status, and battery level. You can use the Dashboard Main VI during robot testing to receive live feedback from the robot. The Dashboard Main VI also displays user-defined data and the current input and output values from the CompactRIO device, the robot, and the driver station.

**Rebuilding the FRC Dashboard Project**

After you customize the FRC Dashboard Project, you must rebuild the project so that the Driver Station launches the most recent version of the project.

Complete the following steps to rebuild the project and specify which project the Driver Station launches automatically.

1. In the **Project Explorer** window of the FRC Dashboard Project, expand the list of Build Specifications to reveal the FRC PC Dashboard application.
2. Right-click **FRC PC Dashboard** and select **Properties** from the shortcut menu. If the **Application Builder Information** dialog box appears, click **OK**.
3. Verify that the **Target filename** is `Dashboard.exe`. The Driver Station reads this executable file as the file to launch.
4. Verify that the **Destination directory** is where you want to save the file. Click **Build** to build the new project.
5. Explore to where you saved the new executable file. Copy the new Dashboard.exe file over to the `C:\Program Files\FRC Dashboard` directory.

Refer to Tutorial 7—Integrating Examples into Robot Code for information about modifying the dashboard project.

Table of Contents

## Tutorial 7—Integrating Examples into Robot Code

Use the FRC examples to better understand how to read robot sensors or control robot actuators. This tutorial uses the Gyro Example to demonstrate how to incorporate example program code into your robot code.

**Note** If you are not familiar with the Team Code VIs such as the Begin VI and Teleop VI, refer to Tutorial 5: Editing Team Code VIs before completing this tutorial about integrating examples.

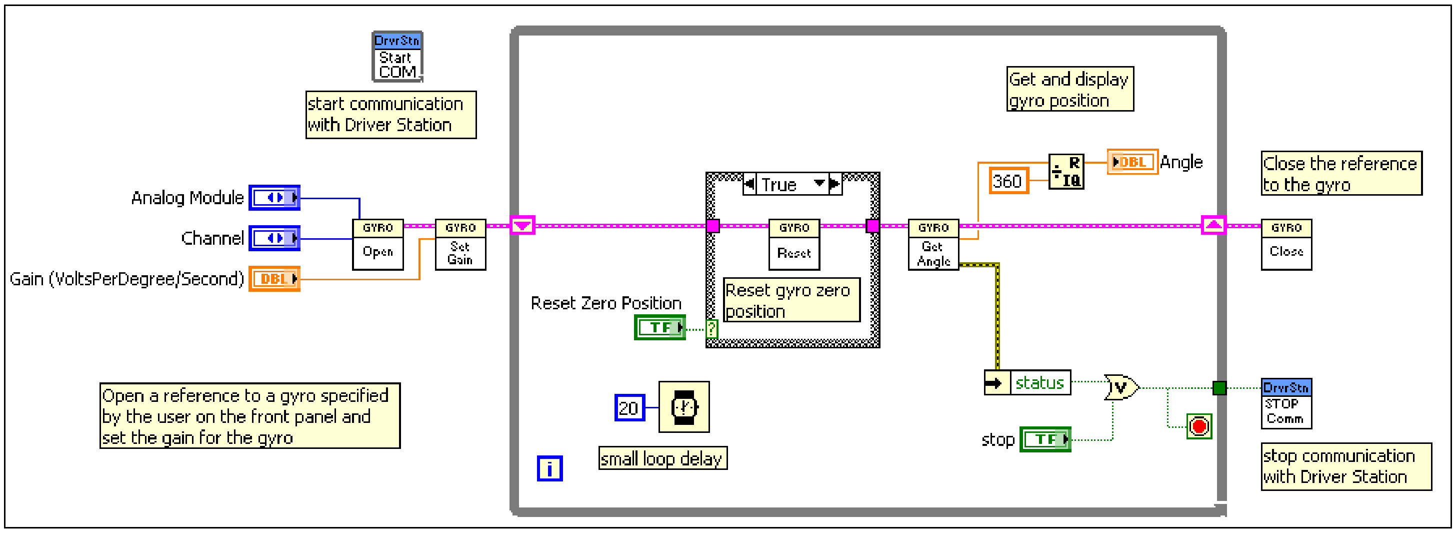
By default, the robot project does not acquire gyroscope data, and the dashboard does not display gyroscope data. Complete the following procedures to integrate the Gyro Example into your robot code so that the robot acquires gyroscope (gyro) data and sends the data to the dashboard. You can then edit the dashboard code to display the data.

**Part 1—Integrating the Gyro Example Code to Read Gyro Data**

First, you must open the example you want to use. Complete the following steps to open the Gyro Example.

1. On the **Getting Started** window, click the **Find FRC Examples** link to display the NI Example Finder.
2. Under the **FRC Robotics** folder, double-click the **Sensors** folder to view sensor examples.
3. Double-click **Gyro Example.lvproj** to open the example project.
4. In the **RT CompactRIO Target (10.0.0.2)** section, double-click the **Gyro Example.vi** item to open the VI.

Notice that the front panel of the VI shows the module and channel the VI uses, as well as how to wire a gyro to your analog breakout board. If you have not used a gyro before, first ensure that you can run this example VI successfully. The block diagram should appear similar to the following figure.

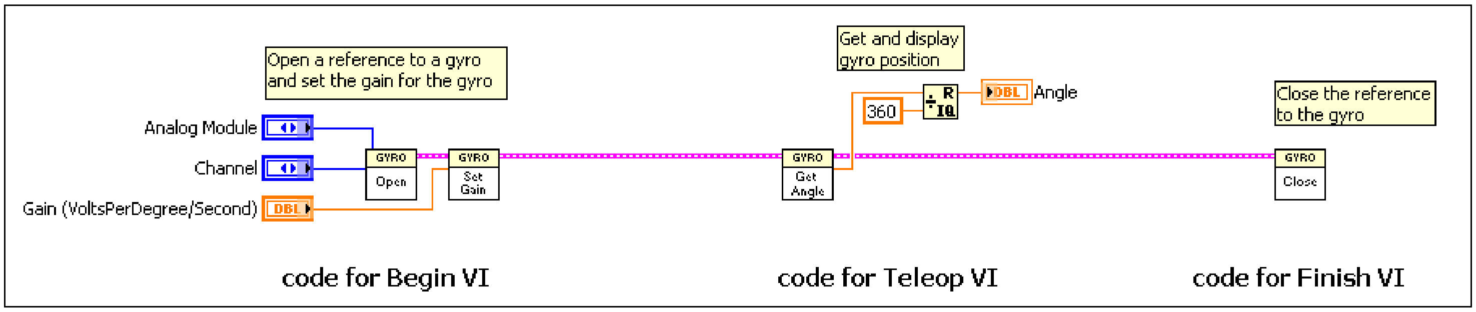

DrvrStn Start COM
start communication with Driver Station
Analog Module
Channel
Gain (VoltsPerDegree/Second)
GYRO Open
GYRO Set Gain
Open a reference to a gyro specified by the user on the front panel and set the gain for the gyro
True
GYRO Reset
Reset gyro zero position
Reset Zero Position
20
small loop delay
Get and display gyro position
360
Angle
GYRO Get Angle
status
stop
Close the reference to the gyro
GYRO Close
DrvrStn STOP Comm
stop communication with Driver Station

To integrate the example code into the robot code, you need to understand what each section of the code does. In general, everything to the left of the loop is for configuration and belongs in the Begin VI. Everything inside the loop reads data, performs calculations and belongs in the Teleop or Autonomous VI. Finally, everything to the right of the loop stops and releases resources and belongs in the Finish VI.

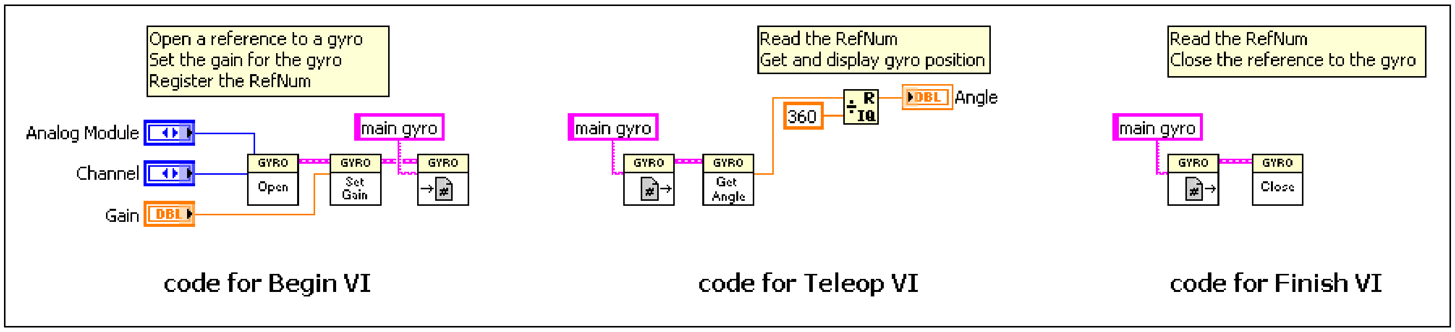
Before you integrate the example code, you can remove the robot code that you no longer need. You can remove the following VIs:

- Because the robot template handles driver station communication, you can remove the Start Communication VI, , and the Stop Communication VI, .
- Because the robot template handles looping and timing, you do not need the While Loop, **stop** terminal, or loop delay.
- Because the robot template does error handling, you do not need the **status** output.
- For simplicity you also can remove the gyro reset code. However, after you get all of the new code working, you may want to add the gyro reset code back. You also can control the reset with a joystick button.

The following figure shows the code you still need to integrate.


Open a reference to a gyro and set the gain for the gyro
Get and display gyro position
Close the reference to the gyro
Analog Module
Channel
Gain (VoltsPerDegree/Second)
DBL
GYRO Open
GYRO Set Gain
GYRO Get Angle
GYRO Close
360
Angle
code for Begin VI
code for Teleop VI
code for Finish VI

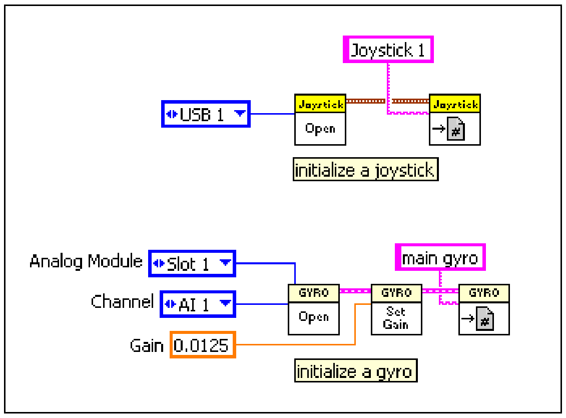
You split the remaining code into the Begin, Teleop, and Finish VIs of your robot code. However, notice there is a common wire among all three subVIs—the RefNum wire. The Begin VI creates the RefNum wire, and you want the RefNum in the Teleop and Finish VIs. You need a way to pass the RefNum between these VIs without the wire. To use the RefNum in all three VIs, use the RefNum Registry Set VI, , and the RefNum Registry Get VI, . You can find these VIs on each WPI Robotics Library palette. If you use the RefNum Registry VIs, you can now remove the RefNum Wire, as shown in the following figure.



Notice that the RefNum Registry VIs require a refnum name. In this example, the name is main gyro. You can choose any name you want for the RefNum Registry Set VI, but you must ensure that you wire the exact same name to the RefNum Registry Get VI. At this point each section of the example code is independent and ready for integration into your robot code. The following sections describe how this code might look in the Begin, Teleop, and Finish VIs.

**Begin VI**

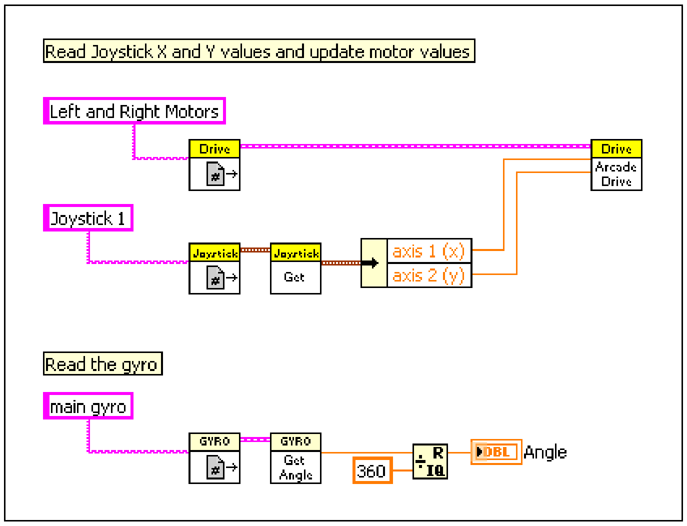
First, you can add the gyro configuration code to the block diagram of the Begin VI. Add the code below the existing joystick configuration code. Your block diagram should appear similar to the following figure.



Notice that you must replace the control terminals with constant values. You do not have to change these values after you wire the physical components of the robot. Ensure that the value of the **Gain** constant matches your specific gyro sensor.

**Teleop VI**

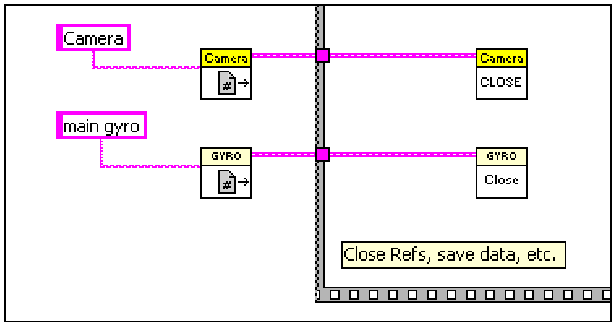
Second, you can edit the code to read the gyro angle. Add the code below the existing joystick and drive code. Your block diagram should appear similar to the following figure.



The **Angle** indicator is not useful in the Teleop VI because you cannot view the front panel of the Teleop VI while the robot is running. Instead, you can send the angle data to the dashboard where you can see the data while the robot is running so that you can use the angle data to help you drive the robot.

**Finish VI**

Finally, you can add the gyro close code below the existing camera close code to stop and release resources after the code runs. The block diagram should appear similar to the following figure.



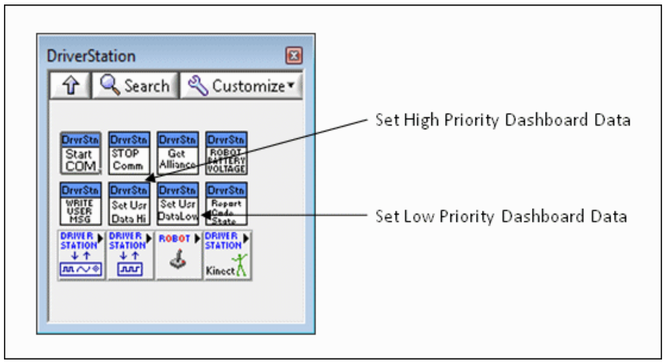
The code does not call the Finish VI during a competition because the Field Management System (FMS) disables all robots at the end of a competition, and then teams typically reboot the robots. Therefore, you do not need to use the Finish VI to stop and release resources during a competition. However, when you develop code and test the robot, you want to call the Finish VI to stop and release resources after each time you run the code. You can call the Finish VI by pressing the **Finish** button in the Robot Main VI. Using the Finish VI and the **Finish** button allow you to run the Robot Main VI multiple times without rebooting the robot each time.

At this point the robot code includes the gyro example code. You can build and deploy the code to use the gyro. Similarly, you can integrate other example VIs by following these same basic steps.

**Part 2—Edit the Robot Code to Send the Gyro Data to the Dashboard**

In Part 1 you integrated the gyro code into the robot code so that the robot code read the gyro data. Next, you can edit the robot code to display the gyro data, such as the angle, on the dashboard.

The **DriverStation** palette includes two VIs that you can use to pass data from the robot to the dashboard—the WPI_DriverStationSet High Priority Dashboard Data VI and the WPI_DriverStationSet Low Priority Dashboard Data VI. The following figure shows these two VIs on the **DriverStation** palette.



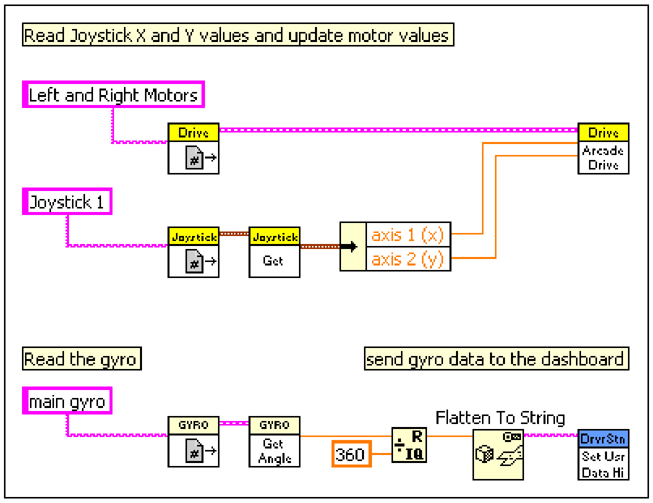
In the robot code, you must use only one instance of each of these VIs. By default, the Build Dashboard Data VI, which is a robot Team Code VI, already uses the WPI_DriverStationSet Low Priority Dashboard Data VI in Build Dashboard Data VI. Therefore, you can use the WPI_DriverStationSet High Priority Dashboard Data VI. At this point, the robot code can read the gyro angle in the Teleop VI, so you can send the gyro data from the Teleop VI to the dashboard. Complete the following steps to send the gyro data to the dashboard.

1. Right-click in a blank space on the Teleop VI block diagram and navigate to the WPI_DriverStationSet High Priority Dashboard Data VI on the **WPI Robotics Library»Driver Station** palette. Add the WPI_DriverStationSet High Priority Dashboard Data VI to the right of the **Angle** terminal on the block diagram of the Teleop VI.
2. The Set Priority Dashboard Data VIs accept only string data, so you must use the Flatten To String function, , to convert the scalar data to

string data. Right-click in a blank space on the Teleop VI block diagram and navigate to the Flatten To String function on the **Programming»Numeric»Data Manipulation** palette. Add the Flatten To String function to the right of the **Angle** terminal on the block diagram of the Teleop VI.

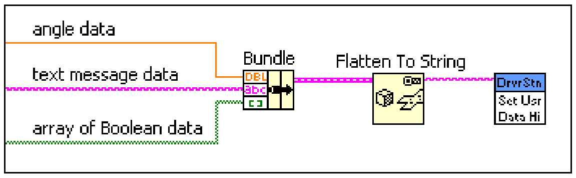
3. Delete the Angle terminal, and then wire the broken wire to the **anything** input of the Flatten To String function.
4. Wire the **data string** output of the Flatten To String function to the **User Data** input of the WPI_DriverStationSet High Priority Dashboard Data VI.

The block diagram should appear similar to the following figure.



At this point, the robot code reads the gyro data and then sends the gyro angle to the dashboard each time you call the Teleop VI.

You also can send multiple data values to the dashboard by wiring a cluster of data to the **anything** input of the Flatten To String function. A cluster can contain any combination of different data types, and you can use the Bundle function to create a cluster. The following figure shows an example of how you can create a cluster to send multiple data values to the dashboard.



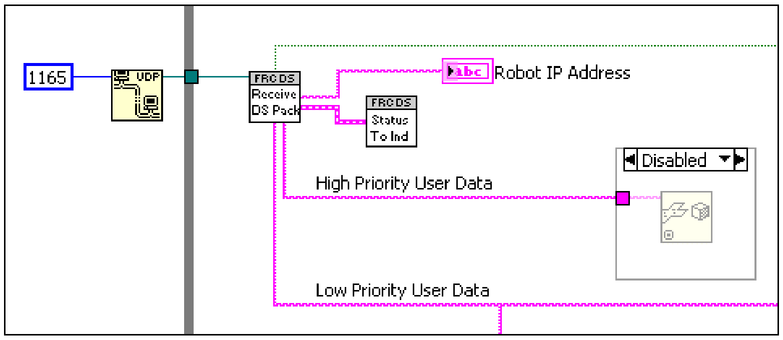
Notice that you can send any data to the dashboard. In Part 3 of this tutorial you edit the code to read the gyro data in Dashboard Main VI.

**An Alternative Way to Send Gyro Data to the Dashboard**

You also can send gyro data to the dashboard by editing `DashboardEnables.ctl` in the Build Dashboard VI so that the control includes the gyro data type. Then you can send the gyro data in the Build Dashboard Data VI and retrieve the gyro data from the **Low Priority User Data** output of the Receive DS Packet VI within the Dashboard Main VI.

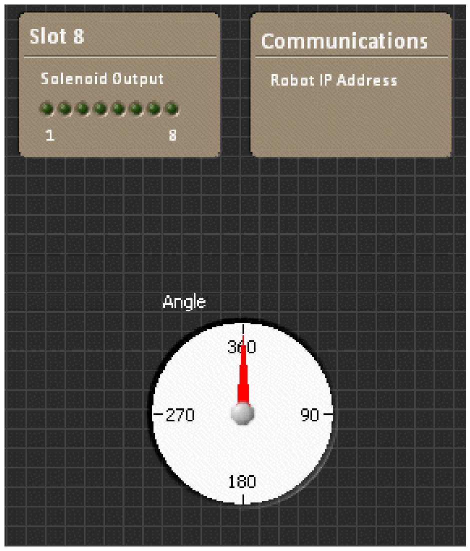
**Part 3—Editing the Dashboard to Display the Gyro Angle While the Robot is Running**

This section assumes that you understand Tutorial 6—Creating a Custom Dashboard. From within the Dashboard Project, open the Dashboard Main VI and look at the block diagram. In the second While Loop from the top, locate the Receive DS Packet VI. Notice the High Priority User Data wire, as shown in the following figure.



This wire contains the gyro angle data while the robot is running. Complete the following steps to edit the code to convert the string data to scalar data and display the gyro angle on the dashboard.

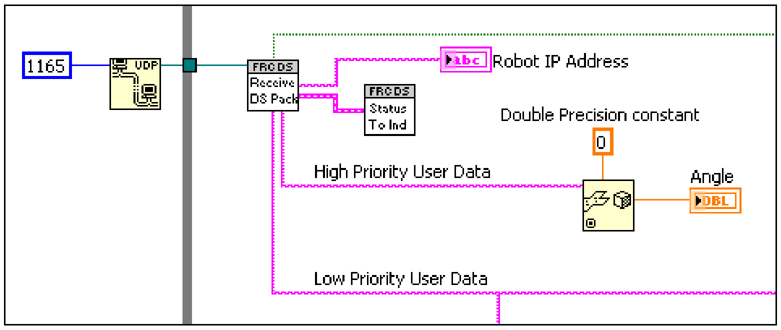
1. Right-click the edge of the Diagram Disable Structure and select **Remove Diagram Disable Structure** from the shortcut menu. Removing this structure allows you to use the Unflatten From String function, .
2. Display the dashboard, which is the front panel of Dashboard Main VI.
3. Right-click a blank space on the dashboard and navigate to the gauge indicator on the **Modern»Numeric** palette.
4. Edit the gauge by changing the range to 0-360, changing the label text from **Gauge** to **Angle**, and changing the label text color to white. The gauge should look similar to the following image.



5. Right-click the gauge and select **Find Terminal** from the shortcut menu.
6. Display the block diagram, and move the **Angle** terminal you found in the previous step to the right of the Unflatten From String function.

   To use the Unflatten From String function, you must wire a value to the **type** input. The type must match the data type you wired to the **anything** input of Flatten To String function in the Teleop VI. That data type is a double precision scalar.
7. To find a double-precision scalar constant, right-click the **Angle** terminal and select **Create»Constant** from the shortcut menu.
8. Delete the wire between the constant and the **Angle** terminal.
9. Position this constant above the Unflatten From String function, and wire the constant to the **type** input.
10. Wire the **value** output of the Unflatten To String function to the Angle terminal.

The block diagram should appear similar to the following block diagram.



Now, you can rebuild and deploy the robot project as shown in Tutorial 4—Developing a Robot Project, and rebuild the dashboard project as shown in Tutorial 6—Creating a Custom Dashboard, to see the gyro angle in the dashboard while you drive you robot.

Table of Contents

## Tutorial 8—Integrating Vision into Robot Code

Use the FRC Vision examples to better understand how to use the camera and the Vision software to find and track objects. This tutorial demonstrates how to incorporate the Rectangular Target Processing example into your robot code.

**Note** This example project assumes the following:

- You have a ring light mounted to the camera.
- You have used the camera setup utility to set the camera address to `10.te.am.11`.
- You have set the laptop IP to `10.te.am.06` or a compatible IP address.

### Part 1—Open the Example

First you must open the example you want to use. Complete the following steps to open the Rectangular Target Processing example.

1. On the **Getting Started** window, go to the **Support** page and click **Find FRC Examples** to display the NI Example Finder.
2. Under the **FRC Robotics** folder, double-click the **Vision** folder to view Vision examples.
3. Double-click **Rectangular Target Processing.lvproj** to open the example project.
4. In the **My Computer** section of the **Project Explorer**, double-click **Rectangular Target Processing.vi** to open the VI.

   Notice that there is also a Rectangular Target Robot Main VI in the **RT CompactRIO Target** section. You will use that VI to deploy the code to your CompactRIO device in part 2, but right now you just want to run and experiment with the example on the host computer.
5. Change the Camera IP address and the Camera Type to match your camera.
6. Connect the camera to the switch or directly to the computer. Then click the **Run** button to run the VI.
7. In the original color image on the left side, click the colored band to see the Processed Image update to show the mask. You should see a live image from your camera. If you do not, make sure that you have configured your camera and PC network correctly with the camera setup utility.

If necessary, click on less saturated areas of the original image until you obtain a good mask. You can also adjust the values in the Processing tab.

8. Measure the distance to the target and compare to the value reported in the Target Info section of the VI.

**Part 2—Run the VI on the RT CompactRIO Target**

Complete the following steps to run the example Vision VI on the CompactRIO device.

1. Close the Rectangular Target Processing VI that you opened from the **My Computer** section. Do not save any changes that you made to the VI.
2. In the **Project Explorer**, right-click **RT CompactRIO Target** and select **Properties** from the shortcut menu. Change the IP Address to match your CompactRIO device and click **OK**.
3. In the **RT CompactRIO Target** section, double-click **Rectangular Target Robot Main.vi** to open the VI.
4. Connect the CompactRIO device and the computer to the switch, and click the **Run** button to run the VI on the RT target.
5. From the block diagram of the Rectangular Target Processing VI, double-click the Vision Processing VI, and then repeat the steps in part 1 to experiment with the VI when it runs on the RT target.
6. Close all the VIs from the example Vision project. Do not save any of the changes you made.

**Part 3—Integrate the Example VI into a Robot Project**

After you experiment with the example and see how it works, you can save a copy of the VI and all the files it depends on to a new location to use it in your robot project.

1. Open the Rectangular Target Robot Main VI from the **RT CompactRIO Target** section in the **Project Explorer**, and open the Vision Processing VI.
2. From the open example VI, select **File»Save As** and select the **Duplicate hierarchy to new location** option. Then click **Continue**.
3. Navigate to your robot project directory and click the **Current Folder** button to save a copy of the example Vision VI and all of its dependency files to your robot project.

   **Note** After you click **Current Folder**, a dialog box appears warning you that you are overwriting two VIs in your robot project, the existing Vision Processing VI and the Robot Globals. This is correct. You do want to overwrite these two existing VIs with the example VIs you are saving.
4. Run your robot project code.
5. From the block diagram, open the Vision Processing VI and verify that it works as expected.

Table of Contents

## Tutorial 9—CAN Jaguar Fundamentals

**Deciding when to use CAN Jaguar**

Before using CAN with Jaguar motor controllers, it is useful to decide why you want to use CAN Jaguar and if it makes sense to do so. Since CAN Jaguar is often an alternative to PWM, this tutorial compares the two and discusses pros and cons of each.

**Using PWM**

The main advantage of PWM motor control is dedicated bandwidth for each motor you control. This means you can add more motors without affecting performance.

**Using CAN Jaguar**

The main advantage of using CAN Jaguar is that it is better for closed loop control. CAN Jaguar can take advantage of internal sensors and directly integrate supported external sensors.

**Understanding CAN Jaguar**

The CAN 2.0B bus used by Jaguar is a 1 megabit bus, which means it transfers one million bits of information each second. Because Jaguar uses Extended Frame Format, each CAN message can be up to 128 bits in length. A single command takes two CAN messages, one ack, which acknowledges the packet was received and one set, which sets the ack. The theoretical rate for commands is calculated as:

$$\frac{(1{,}000{,}000\ \text{bits/second})}{(128\ \text{bits/message})(2\ \text{messages/command})}$$

This result is 3906 commands/sec. This result is theoretical because it does not account for some bus overhead due to bit stuffing and inter-frame spacing which reduces the effective bandwidth. The CompactRIO does not have a CAN bus interface. You must use a bridge device (that communicates over another interface that the CompactRIO supports) to access the CAN bus. These bridge devices cause more overhead and increase the latency to access the CAN bus. If you are using the Black Jaguar Serial bridge, the RS-232 bridge bus rate is calculated as:

$$\frac{(115{,}200\ \text{symbols/second})}{(10\ \text{symbols/byte})(14\ \text{bytes/message*})}$$

*Up to 26 bytes/message based on packing (odds are <1% per byte)

This typically equals around 822 messages/second. The RS-232 bus is full duplex so the typical command rate is 822 commands/second. Ethernet has higher bandwidth than the CAN bus but also higher latency, so achieving max rate on CAN is still a challenge based on the implementation of the bridge you use. On top of these theoretical limits, there is also overhead introduced by the software in the CompactRIO device, the bridge device, and the Jaguar motor controller. Rates close to these limits can only be achieved if code is pipelined to saturate the bus. This means that the code sending a Set Output for one CAN Jaguar must not be blocked by the code waiting for an ack from another CAN Jaguar. LabVIEW does this well because parallel programming is easy and natural in LabVIEW. However, in general it is good to remember that CAN is high latency compared to direct FPGA access. CAN is a serial bus with a longer, slower path. You cannot send a new message to a Jaguar until the ack has been received for the last message to that same Jaguar.

CAN Jaguar controllers have built-in sensors – one for current, one for temperature, and provisions for automatic limit switch handling. You can only access these sensors using CAN. The Jaguar also has terminals for external sensors, including a potentiometer and a Quadrature encoder, which can only be accessed using CAN.

Jaguar can do closed loop updates at a rate of 1000 updates/second. This is better for control loops compared to PWM with PID on the CompactRIO device, which sends 200 updates/second to Jaguar and 100 updates/second to Victor. However, the onboard Jaguar control is limited to the internal PID algorithm and supported sensors. This means it is faster, yet less flexible than a control loop on the CompactRIO device.

CAN Jaguar allows you to read the status of many parameters. These parameters include Output Voltage, Bus Voltage, Current, Temperature, Position, Speed, Limit Switches, Faults, and Power Cycle. While these parameters are useful, it is important to only read what is needed, and to only read them when needed. Remember, the query for this information, as well as the data returned, all share the same CAN bus.

**Sample Use Cases**

Here are a few use cases using the information above to determine whether it is best to use CAN Jaguar, PWM, or a combination of the two.

**Use Case 1: You have 10 open loop motors and you want to simplify your wiring.**

CAN Jaguar needs to update each motor as fast as or faster than PWM, and with all 10 motors sharing the bandwidth of a single bus. For open loop control use PWM where each motor gets a dedicated bus. The effective bandwidth of 10 PWM channels to 10 Jaguars is 200 updates/second * 10 channels = 2000 commands/second equivalent.

**Use Case 2: You have 4 open loop drive motors and 3 closed loop arm control motors.**

Use a combination of PWM and CAN Jaguar. Use PWM for the open loop drive motors. Each can have a dedicated bus and not use any of the CAN bus bandwidth. You also have the choice of using Victors instead of Jaguars (your stock of motor controllers may be a mixture of Jaguars and Victors). Use CAN Jaguar for the closed loop arm controllers. Use an encoder wired to the Jaguar and the CAN Jaguar PID to perform closed loop monitoring. Since the Jaguar is handling the control, you only need to send set points. These can be sent at a much slower rate than if you are directly setting motor output. With the control loop running on the Jaguar, you free up CompactRIO CPU usage. This can improve overall robot performance.

**Use case 3: You need to add another motor to use case 2. The motor will run open loop, but you need to read status from the motor. In this case you need to know the current draw of the motor to make sure it isn't overloaded.**

You can use PWM, add a donut current transformer on one motor lead, add a small circuit, and read voltage across a resistor (or wire an in-line Hall-effect sensor) to accomplish this. However, it is easier to use CAN Jaguar since it allows you to read Current as a status item. Depending on your requirements, an external current sensor can have higher precision than is available with the internal current sensor and may still be more desirable. With only three closed motors on the CAN bus, adding one open loop motor is reasonable.

The solution provided for each use case is not the only solution. What is important is to figure out when it is best to use CAN Jaguar, and that the best solution may be a combination of PWM for some motors and CAN Jaguar for others. For information on programming specifics, see the CAN Jaguar examples in the NI Example Finder.

Refer to the NI FRC Community website for a CAN Getting Started tutorial, which provides an introduction to setting up the CAN Jaguar hardware, and a LabVIEW CAN Jaguar API Introduction, which provides information on the CAN Jaguar functions available in LabVIEW.

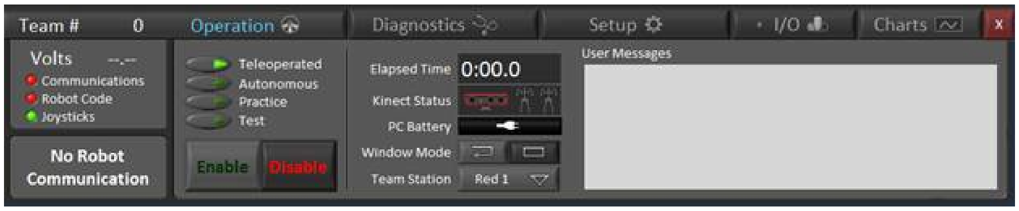
Table of Contents

## Tutorial 10 - Robot Simulation

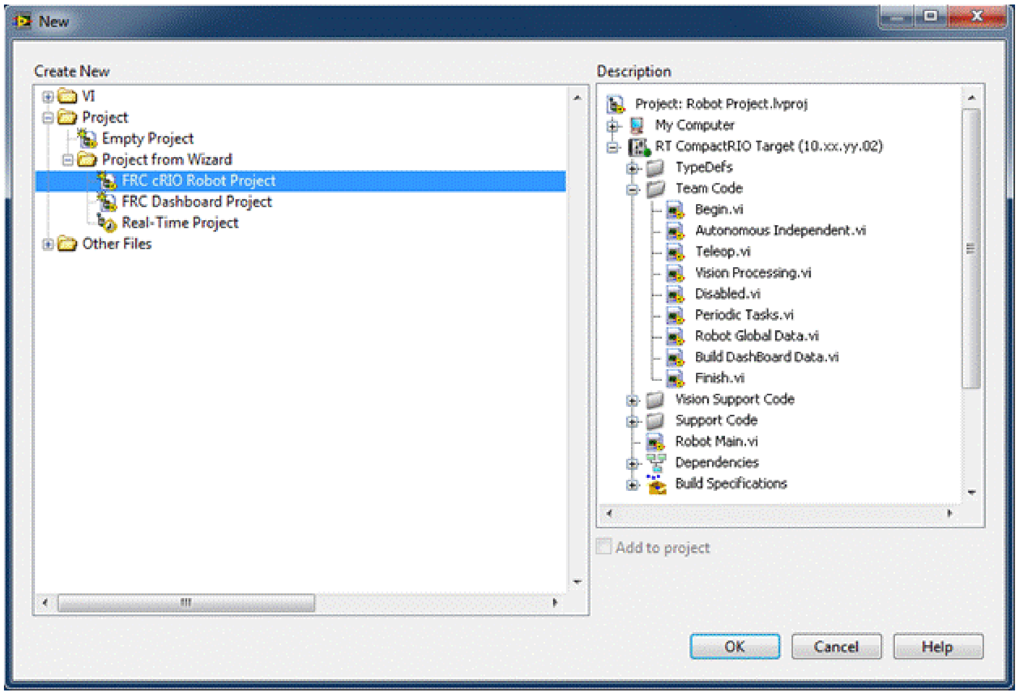
With LabVIEW you can use Robot Simulation to program a predefined robot without having an RT CompactRIO Target. This allows multiple developers to concurrently create and test LabVIEW code without requiring each developer to have access to the hardware. Programming is the same, except only the predefined Actuators and Sensors on the simulated robot are supported. Robot code that has been developed and works in simulation mode can be moved to the RT CompactRIO Target and run on a real robot.

**Opening the Robot Simulator**

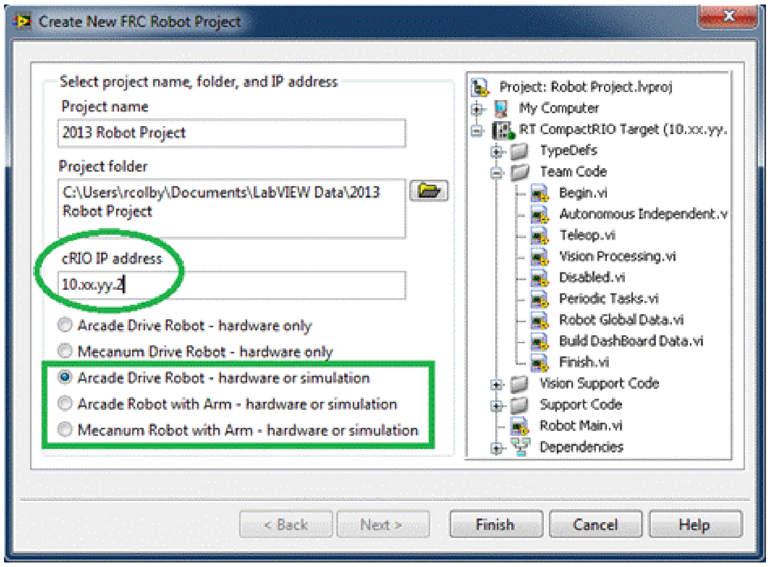
1. Start the FRC Driver Station. This is required for both real and simulated robots.



2. Create a New **FRC cRIO Robot Project** either from the Getting Started Window or by going to File>>New…



3. Update the **cRIO IP address** based on your Team Number, and choose one of the options that includes **hardware or simulation**. Note that you can choose one of the **hardware only** options if you are not using the Simulator and would like the project files to load quicker.


Create New FRC Robot Project

Select project name, folder, and IP address

Project name: 2013 Robot Project

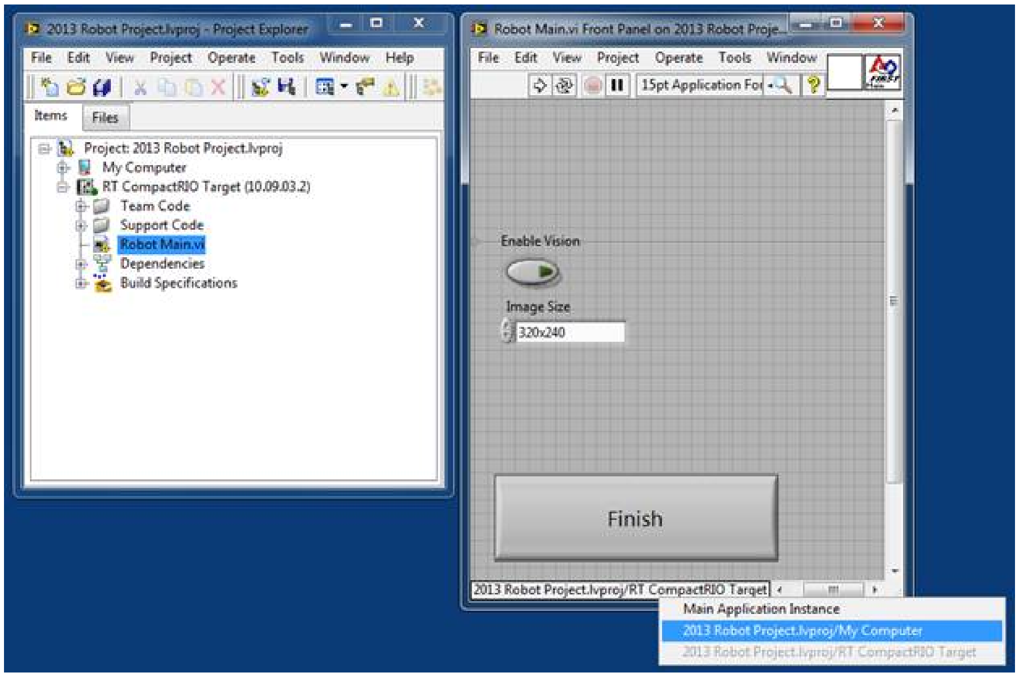
Project folder: C:\Users\rcolby\Documents\LabVIEW Data\2013 Robot Project

cRIO IP address: 10.xx.yy.2

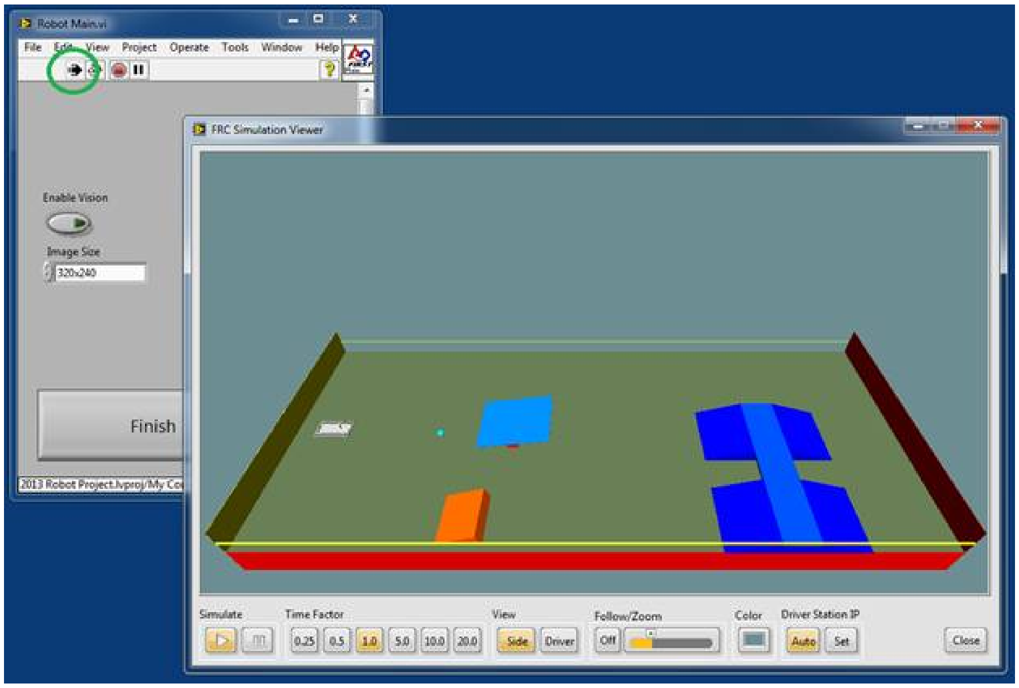
- Arcade Drive Robot - hardware only
- Mecanum Drive Robot - hardware only
- [✓ Arcade Drive Robot - hardware or simulation]
- Arcade Robot with Arm - hardware or simulation
- Mecanum Robot with Arm - hardware or simulation

4. When the LabVIEW Project opens, select and open **Robot Main.vi**.

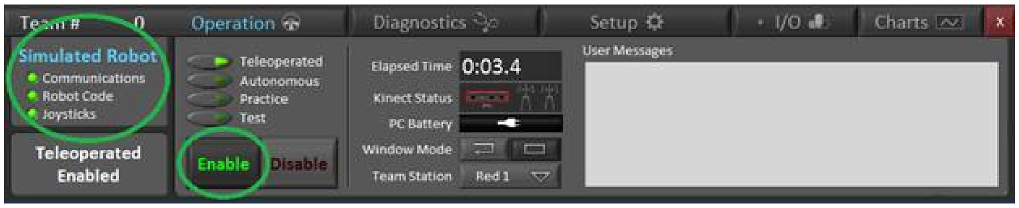
5. Right-click in the lower left corner of **Robot Main.vi** and choose **2013 Robot Project.lvproj/My Computer**. Wait a moment while the subVI's reload.


2013 Robot Project.lvproj - Project Explorer
Project: 2013 Robot Project.lvproj
My Computer
RT CompactRIO Target (10.09.03.2)
Team Code
Support Code
Robot Main.vi
Dependencies
Build Specifications
Robot Main.vi Front Panel on 2013 Robot Proje...
Enable Vision
Image Size
320x240
Finish
2013 Robot Project.lvproj/RT CompactRIO Target
Main Application Instance
2013 Robot Project.lvproj/My Computer
2013 Robot Project.lvproj/RT CompactRIO Target

6. Run **Robot Main.vi** and the **FRC Simulation Viewer** opens.

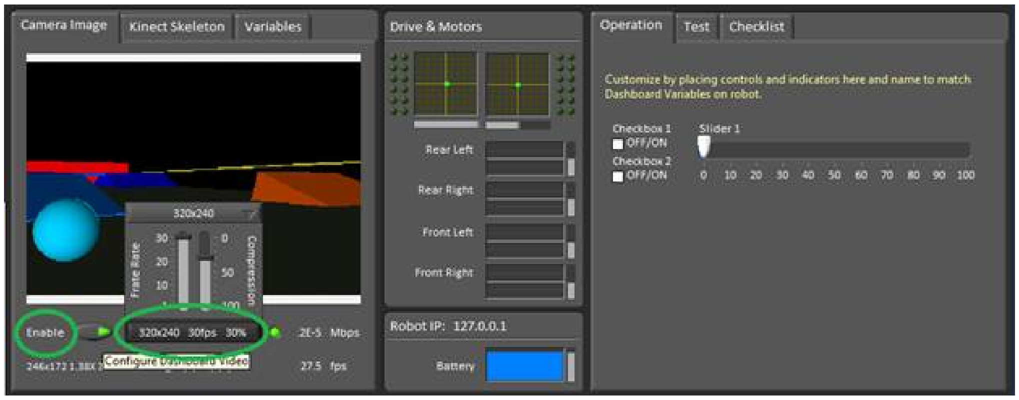


7. The **FRC Driver Station** should now show that you have a **Simulated Robot**. Click **Enable** to place the robot in **Teleoperated Enambled** mode.

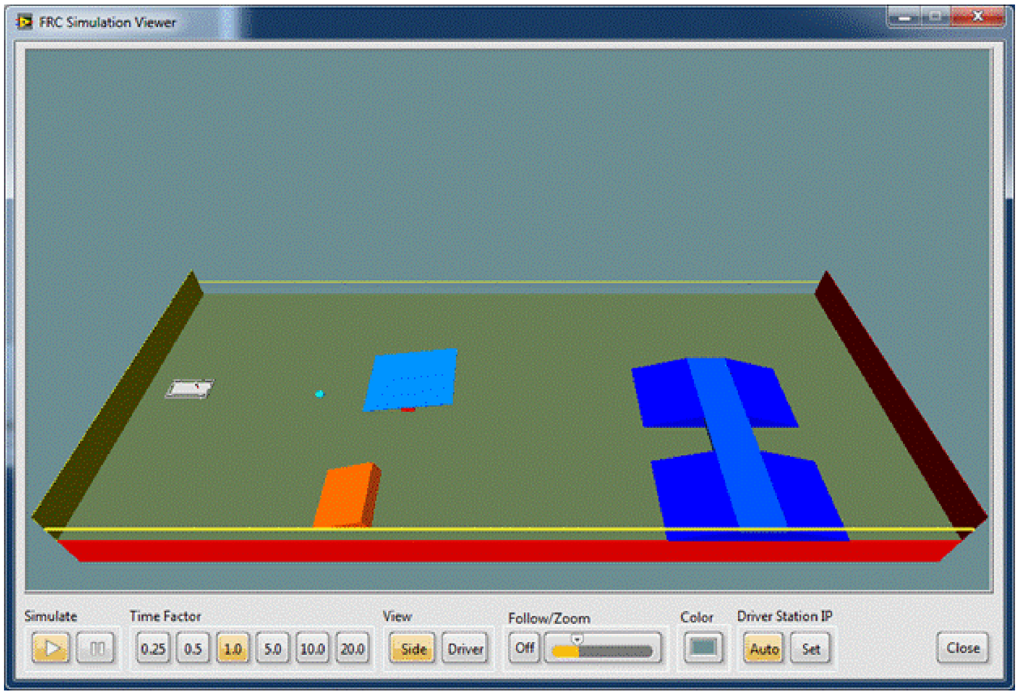


8. You should now be able to use a joystick to drive the robot in the **FRC Simulation Viewer**.

9. The **Camera Image** and **Drive & Motors** can be monitored in the **FRC PC Dashboard**.  Click **Enable** under **Camera Image** to view the Camera, and left-click the resolution to change the **Frame Rate**.  Changing to a lower frame rate will help performance on slower computers.



**Using the Simulator Viewer**



Aside from Teleoperation of the simulated robot using the joystick, there are several options in the FRC Simulator Viewer to help customize your simulation environment:

- **Simulate**—**Run** and **Pause** buttons are available.
- **Time Factor**—allows you to run the dynamic model of the system slower than real time (minimum 0.25) and faster than real time (maximum 20.0). A Time Factor of 1.0 will be representative of the system in real time.
- **View—Side** or **Driver** enables you to default back to one of the two traditional points of view. If **Follow** is **Off**, left-clicking and panning in the Viewer window will allow you to choose a custom viewing angle.
- **Follow**—If you turn Follow **On**, the Viewer will keep the robot centered in the screen from whichever **View** you have selected. If you turn Follow **Off**, left-clicking and panning in the Viewer window will allow you to choose a custom viewing angle, but zooming is no longer possible.
- **Color**—selects the background color.
- **Driver Station IP—Auto** should be set to connect with the correct IP Address based on the RT CompactRIO Target properties in the LabVIEW Project Explorer, but there is also an option to **Set** the IP address manually.
- **Close**—Close the FRC Simulation Viewer and stop Robot Main.vi.

**Programming a simulated robot**

The simulated robots are predefined - they cannot be changed. But they are the same as real robots in that they have sensors and actuators attached. By default each project includes code to drive the robot. Beyond that, it is up to you to add code to make use of additional sensors and actuators included on each robot. For example, each simulated robot includes a gyro. By following **Tutorial 7 - Integrating Examples into Robot Code**, you can learn how to add Gyro code for your simulated robot. Once you get that working in simulated mode, the same code can be built for a real robot.

In order to program each simulated robot, you need to know which sensors and actuators it has. You also need to know how each is wired - which module and channel each uses.

Remember that when you create an FRC Robot Project, you have three options that will work with simulation:

- **Arcade Drive Robot** – hardware or simulation
- **Arcade Robot with Arm** – hardware or simulation

- **Mecanum Robot with Arm** – hardware or simulation

The simulated sensors and actuators are dependent on which project you choose. Below is a list for each project. You can also find this list within each project in **Robot Simulation Readme.html** under **My Computer**.

**Supported Actuators and Sensors on the Simulated Robots**

**1. Arcade Drive Robot**

**Actuators on the simulated robot**

1. Left Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 1
2. Right Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 2
3. Camera Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 5
    - Angular Range = 170

**Sensors on the simulated robot**

1. Encoder on Right Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 3
    - B Channel = DIO 4
2. Encoder on Left Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 5
    - B Channel = DIO 6
3. Gyro
    - Analog Module = Analog Module 1
    - Analog Channel = AI 1
4. Ultrasonic
    - Ping Digital Module = Digital Module 1
    - Ping DIO Channel = DIO 1
    - Echo Digital Module = Digital Module 1
    - Echo DIO Channel = DIO 2
5. AXIS M1011 Camera

**2. Arcade Robot with Arm**

**Actuators on the simulated robot**

1. Left Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 1
2. Right Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 2
3. Camera Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 5
    - Angular Range = 170
4. Arm Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 6
    - Angular Range = 170
5. Gripper Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 7
    - Angular Range = 170

**Sensors on the simulated robot**

1. Encoder on Right Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 3
    - B Channel = DIO 4
2. Encoder on Left Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 5
    - B Channel = DIO 6
3. Gyro
    - Analog Module = Analog Module 1
    - Analog Channel = AI 1
4. Ultrasonic

    - Ping Digital Module = Digital Module 1
    - Ping DIO Channel = DIO 1
    - Echo Digital Module = Digital Module 1
    - Echo DIO Channel = DIO 2
5. AXIS M1011 Camera

**3. Mecanum Robot with Arm**

**Actuators on the simulated robot**

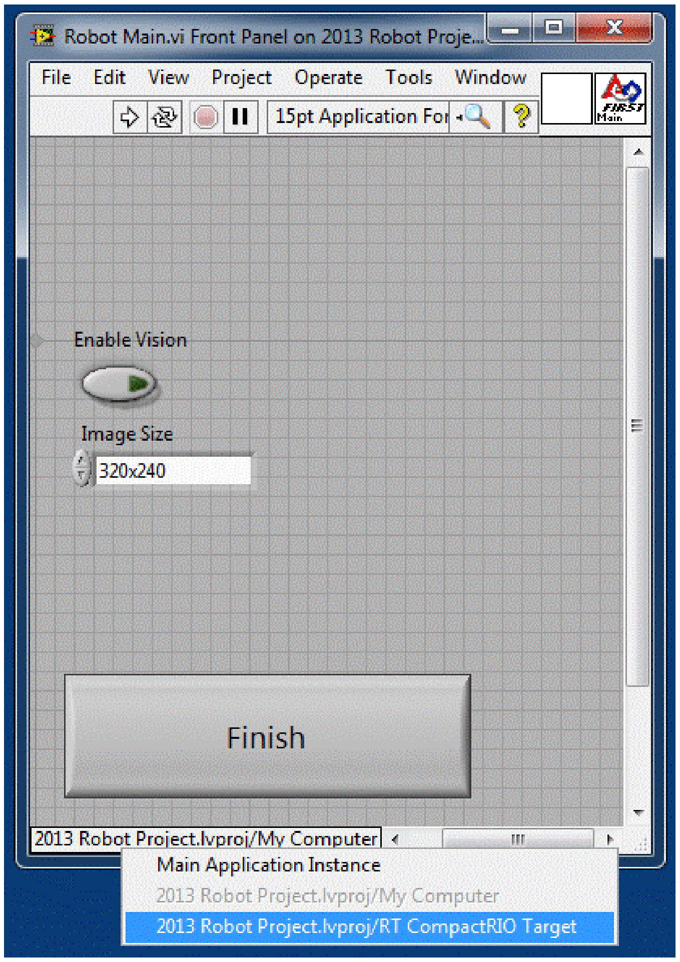
1. Left Front Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 1
2. Right Front Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 2
3. Left Rear Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 3
4. Right Rear Motor
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 4
5. Camera Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 5
    - Angular Range = 170
6. Arm Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 6
    - Angular Range = 170
7. Gripper Servo
    - Digital Module = Digital Module 1
    - PWM Channel = PWM 7
    - Angular Range = 170

**Sensors on the simulated robot**

1. Encoder on Right Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 3
    - B Channel = DIO 4
2. Encoder on Left Motor
    - Digital Module = Digital Module 1
    - A Channel = DIO 5
    - B Channel = DIO 6
3. Gyro
    - Analog Module = Analog Module 1
    - Analog Channel = AI 1
4. Ultrasonic
    - Ping Digital Module = Digital Module 1
    - Ping DIO Channel = DIO 1
    - Echo Digital Module = Digital Module 1
    - Echo DIO Channel = DIO 2
5. AXIS M1011 Camera

**Notes:**

1. If you run a robot in simulation mode, but you have programmed it for I/O not supported in simulation mode, then errors will be generated that slow the simulation performance. You can check for errors under the Diagnostics tab of the FRC Driver Station.
2. The E-Stop (space bar) works in Simulation Mode. To reset it, simply wait 5 seconds and the reset **Robot main.vi**.
3. To change the **Robot Main.vi** to be able to run on the target again, you will need to right click in the lower left corner of **Robot Main.vi** and choose **2013 Robot Project.lvproj/My Computer**. Wait a moment while the subVI's reload.


Robot Main.vi Front Panel on 2013 Robot Proje...
File Edit View Project Operate Tools Window
15pt Application Fo
Enable Vision
Image Size
320x240
Finish
2013 Robot Project.lvproj/My Computer
Main Application Instance
2013 Robot Project.lvproj/My Computer
2013 Robot Project.lvproj/RT CompactRIO Target

---

## Troubleshooting the Robot

In the previous sections, you learned how to program a robot using the CompactRIO device, the NI Robotics CompactRIO projects, and the WPI Robotics Library VIs. However, at times, the robot might not work as you expect. This tutorial provides information about troubleshooting issues that you might encounter when working with the robot and solutions to address those issues.

### TroubleShooting Common Issues

This section introduces you to tools you can use to diagnose and correct common issues.

**Driver Station Diagnostics Page**

Before you try anything else, view the **Messages** on the Diagnostics page of the driver station. If your code returns errors to the driver station, those errors cause your code, and therefore your robot, to run less efficiently.

Hover the cursor over items on the Diagnostics page to view information about possible problems with the robot system and suggestions on how to correct the problems. Edit your robot system or robot code to correct the errors.

**Driver Station Charts Page**

Use the Charts page to learn how your robot uses system resources. This tool can help you determine why your code might run slow and how to improve the performance of your robot code. This tool replaces the Real-Time System Manager that you might have used in previous FRC seasons.

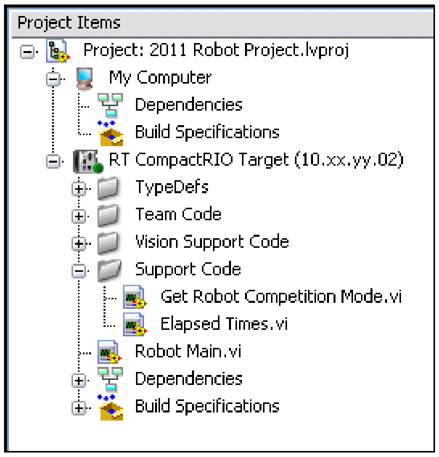
**Profile Performance and Memory Tool**

Use the Profile Performance and Memory tool to see timing statistics and memory usage for your code. Navigate to **Tools»Profile»Performance and Memory** to open this tool.

**Elapsed Times VI**

The Elapsed Times VI,

, is a component of the robot project that you create from the **FRC cRIO Robot Project** wizard. The VI is located in the **Support Code** folder, shown as follows.

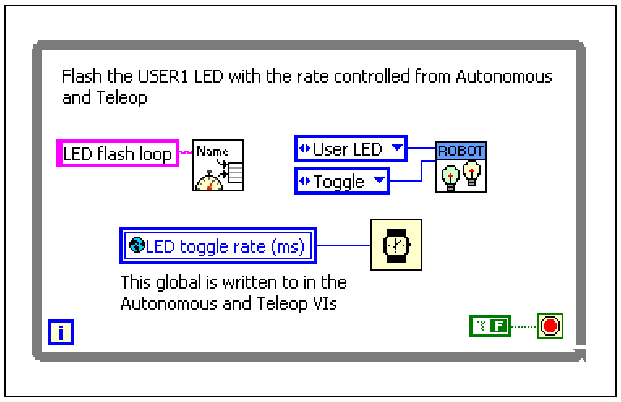

Project Items
- Project: 2011 Robot Project.lvproj
  - My Computer
    - Dependencies
    - Build Specifications
  - RT CompactRIO Target (10.xx.yy.02)
    - TypeDefs
    - Team Code
    - Vision Support Code
    - Support Code
      - Get Robot Competition Mode.vi
      - Elapsed Times.vi
    - Robot Main.vi
    - Dependencies
    - Build Specifications

You can use this VI throughout your robot code as a tool to show you how long a loop or a VI takes to run. Drop the Elapsed Times VI into a loop in your robot code or onto the block diagram of a robot code VI. Then, run your robot code, and open the Elapsed Times VI to see a list of all the locations in which you placed the VI, as well as the amount of time each part of the code takes to run. The Elapsed Times VI includes a **Call Name** input you can use to identify where you put the VI in your code. If you do not wire a value to this input, the Elapsed Times VI uses the name of VI you placed the Elapsed Times VI in.

You can test sections of robot code to make sure the code does not use more CPU time than you expect. For example, you can test the LED flashing loop within the Periodic Tasks VI. Complete the following steps to check whether the loop iterates at the correct speed.

1. Open the Periodic Tasks VI, accessible from the **Team Code** folder of the robot project.
2. Press the <Ctl-E> keys to view the block diagram.
3. From the **Support Code** folder of the robot project, drag the Elapsed Times VI to the USER1 LED loop of the Periodic Tasks VI.
4. Right-click the **Call Name** input of Elapsed Times VI, and select Create»Constant from the shortcut menu.
5. Enter `LED flash loop` in the string constant to specify the name that appears when you open the Elapsed Times VI while the robot code is running.

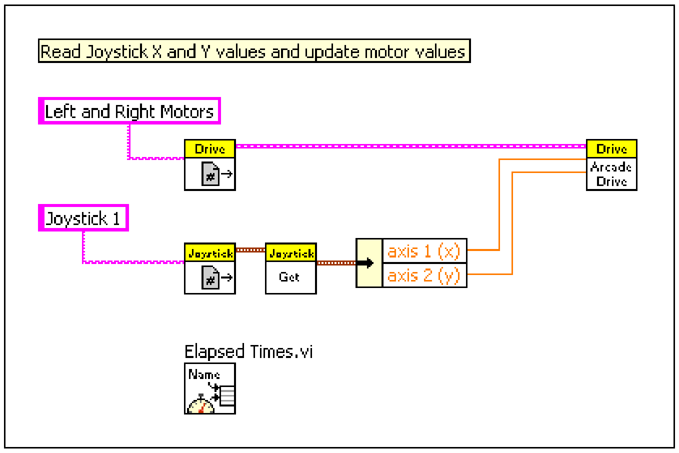
Your robot code should appear similar to the following figure.


Flash the USER1 LED with the rate controlled from Autonomous and Teleop
LED flash loop
Name
User LED
Toggle
ROBOT
LED toggle rate (ms)
This global is written to in the Autonomous and Teleop VIs
i
F

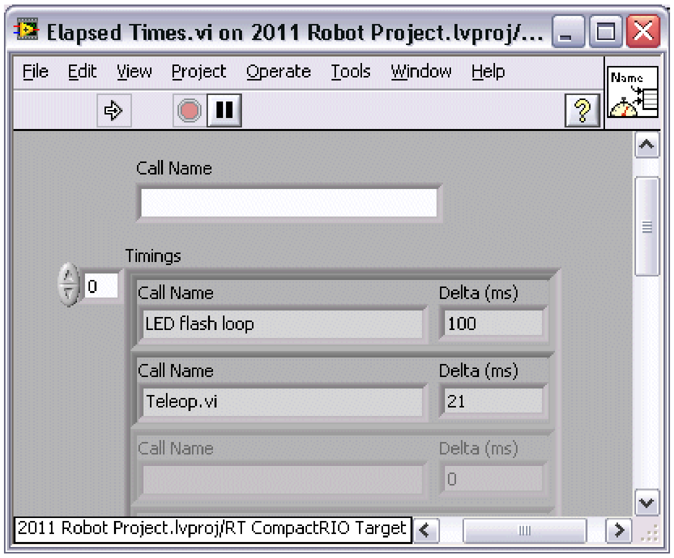
You also can test to see how long the Teleop code takes to run. Complete the following steps to run this test.

1. Open the Teleop VI, accessible from the **Team Code** folder of the robot project.
2. Press the <Ctl-E> keys to view the block diagram.
3. From the **Support Code** folder of the robot project, drag the Elapsed Times VI to the block diagram of the Teleop VI.

The block diagram should appear similar to the following figure.



Notice that you did not wire a value to the **Call Name** input in the previous steps. In this case the Elapsed Times VI automatically uses its caller VI, Teleop, for the call name. While your robot code is running, open the Elapsed Times VI. View the front panel to see a real-time view of how long each section of code takes to run. This information can help you find sections of code that may use more CPU time than you expect.


Elapsed Times.vi on 2011 Robot Project.lvproj/...
File Edit View Project Operate Tools Window Help
Name

Call Name

Timings

| Call Name | Delta (ms) |
|---|---|
| LED flash loop | 100 |
| Teleop.vi | 21 |
|  | 0 |

2011 Robot Project.lvproj/RT CompactRIO Target

**Note** When you finish using the Elapsed Times VI to check your code, remove the VI before you build and deploy your robot code because the Elapsed Times VI also uses a small amount of memory and CPU time.

## TroubleShooting General Issues

The following table lists common issues you might encounter when programming or working with the robot, as well as solutions to these issues.

| Issue | Solution |
|---|---|
| The CompactRIO Imaging Tool does not run on some computers with two Internet connections. | Disable one of the Internet connections and run the CompactRIO Imaging Tool again.<br><br>For example, if a computer has both a wireless Internet connection and a wired connection, disable the wireless connection and run the CompactRIO Imaging Tool again.<br><br>Refer to Tutorial 1—Setting up the CompactRIO for information about configuring the CompactRIO device. |
| The motors on the robot do not run when I attempt to run them. | Use the safety configuration VIs, available for PWM, Relay, Solenoid, RobotDrive, and MotorControl, to ensure that the loop that contains the set output does not run faster than the **timeout (ms)** input value specified in the safety configuration VI. If the speed of the loop is not monitored and another loop starves the loop that contains the set output, the robot may not respond to input values, such as joystick values, and may not stop. The safety configuration VIs are a safety feature to ensure the robot responds to input values.<br><br>To find an example program that uses the safety configuration VIs, click the **Find FRC Examples** link in the **Getting Started** window, and double-click the **FRC Robotics»Safety** folder.<br><br>Refer to the FRC tutorialson the National Instruments *FIRST* Community Web site for information about using the safety configuration VIs. You also can refer to the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.<br><br>If you are using the user watchdog, ensure that you configured the Watchdog VIs to feed the user watchdog. |
| When I try to deploy a program to the CompactRIO device, the program does not run. Sometimes the robot begins operation although I did not run a program. | Launch the **CompactRIO Imaging Tool** dialog box and examine the **Choose Development Environment** section. Ensure that you select the development environment in which you want to work, and then redeploy the program.<br><br>If you specify a development environment different from the one in which you want to work, the program from the other development environment might run instead and prevent you from accessing the CompactRIO device.<br><br>Refer to Tutorial 1—Setting up the CompactRIO for more information about the CompactRIO Imaging Tool. |
| I cannot access the Axis camera using the FIRST Vision VIs. | Ensure the Axis camera is connected to the CompactRIO device using the orange Ethernet crossover cable in the FRC.<br><br>Refer to Tutorial 2—Setting up the Axis Camera for more information about configuring the camera.<br><br>Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use the FIRST Vision VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**. |
| When I try to run the CompactRIO Imaging Tool, the CompactRIO Imaging Tool dialog box does not list any CompactRIO devices connected to the host computer. | Check the network firewall and ensure that the firewall allows the computer to access the CompactRIO device.<br><br>You might need to disable the firewall for the CompactRIO Imaging Tool to run with no errors. |
| When I run the CompactRIO Imaging Tool, the tool stops while downloading an image to the CompactRIO device. | Ensure that the network firewall allows the computer to access the CompactRIO device with both an 0.0.0.0 IP address and an 10.*xx*.*yy*.02 IP address. During the imaging process, the CompactRIO Imaging Tool sets the IP address of the CompactRIO device to 0.0.0.0.<br><br>You might need to disable the firewall for the CompactRIO Imaging Tool to run with no errors. |

Table of Contents

## 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.

Click the **Find FRC Examples** link on the **Getting Started** window to find example programs that use the WPI Robotics Library VIs. You also can access this information in the *LabVIEW Help*, available by selecting **Help»LabVIEW Help**.

**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 figures illustrate how to open a reference to an encoder, start the same encoder, stop the encoder, and then close the corresponding reference.

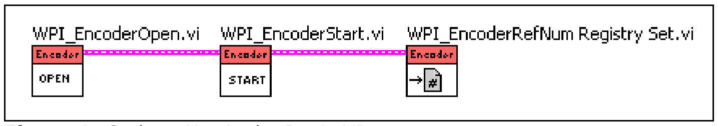


**Figure 1**: Code to Use in the Begin VI

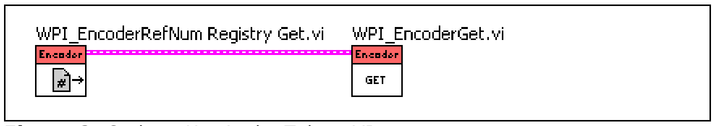


**Figure 2**: Code to Use in the Teleop VI

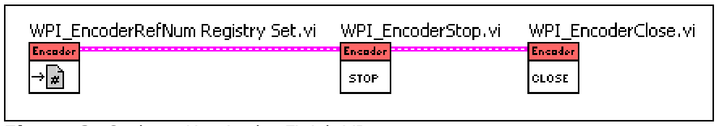


**Figure 3**: Code to Use in the Finish VI

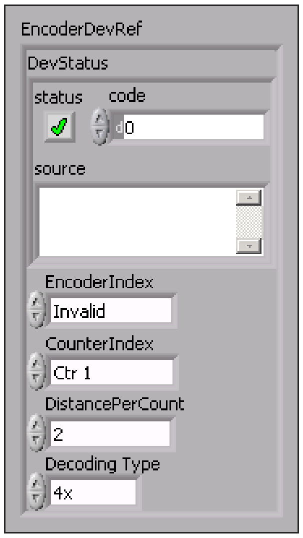
In the Begin VI, the WPI_EncoderOpen VI reserves an available encoder index and returns an **EncoderDevRef** output reference that identifies the encoder. Wire that reference to the **EncoderDevRev** input reference of the WPI_EncoderStart VI to start the encoder. Then, wire the **EncoderDevRef** output reference of the WPI_EncoderStart VI to the **EncoderDevRev** input reference of the WPI_EncoderRefNum Registry Set VI to register the refnum so that other template VIs can use the refnum. For example, in the Teleop VI, you can use the WPI_EncoderRefnum Registry Get VI to read and pass the EncoderDevRef to the WPI_EncoderGet VI to read the current encoder value. Similarly, you can use the refnum in the Finish VI to stop the encoder and release the resources it uses.

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 2011\manuals` directory and opening `FRC_Getting_Started.pdf`, for more information about the LabVIEW error cluster.

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



**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.

When 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.

Table of Contents

---

## Online Forums and Tutorials

Refer to the FIRST Robotics Competition Community on the National Instruments Web site to read and share support information.

Refer to the FRC Training Material and Resources on the National Instruments Web site to access step-by-step instructions for programming the robot.

## Other Help Resources

**Programming in LabVIEW**

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

- *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. Access the *LabVIEW Help* by selecting **Help»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 2011\manuals` directory and opening `LV_Quick_Reference.pdf`.

**Using the CompactRIO Device**

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

- *CompactRIO™ cRIO–9072/3/4 Operating Instructions and Specifications*—This document describes how to install, configure, and use the CompactRIO device for FRC. Access this manual by navigating to the `National Instruments\CompactRIO\manuals` directory and opening `crio-9072_3_4_Operating_Instructions.pdf`.
- *NI 9201/9221 Operating Instructions*—The following 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`.

Table of Contents