

# USING THE FRC FRAMEWORK

## TABLE OF CONTENTS

- Overview.....2
- FRC Robot Project.....2
  - Basic FRC Robot Project.....3
    - Basic Robot Main VI.....4
    - Basic Robot Global VI.....6
    - Autonomous Independent VI .....6
    - ErrorDisplay VI .....7
  - Advanced FRC Robot Project .....7
    - Robot Main VI.....8
    - ErrorDisplay VI .....8
    - Team Code VIs .....8
    - Type Definitions.....11
- Deploying the FRC cRIO Robot Project .....13
  - Deploying the Program Using the Run Button .....13
  - Deploying the Program from the Project Explorer Window.....13
  - Building and Deploying a Stand-Alone Application .....13
  - Connecting to the CompactRIO Device .....14
- FRC Dashboard Project .....15
- Copyright .....17
  - Trademarks.....17
  - Patents.....17

## OVERVIEW

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

## FRC ROBOT PROJECT

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

Complete the following steps to create an FRC robot project.

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

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

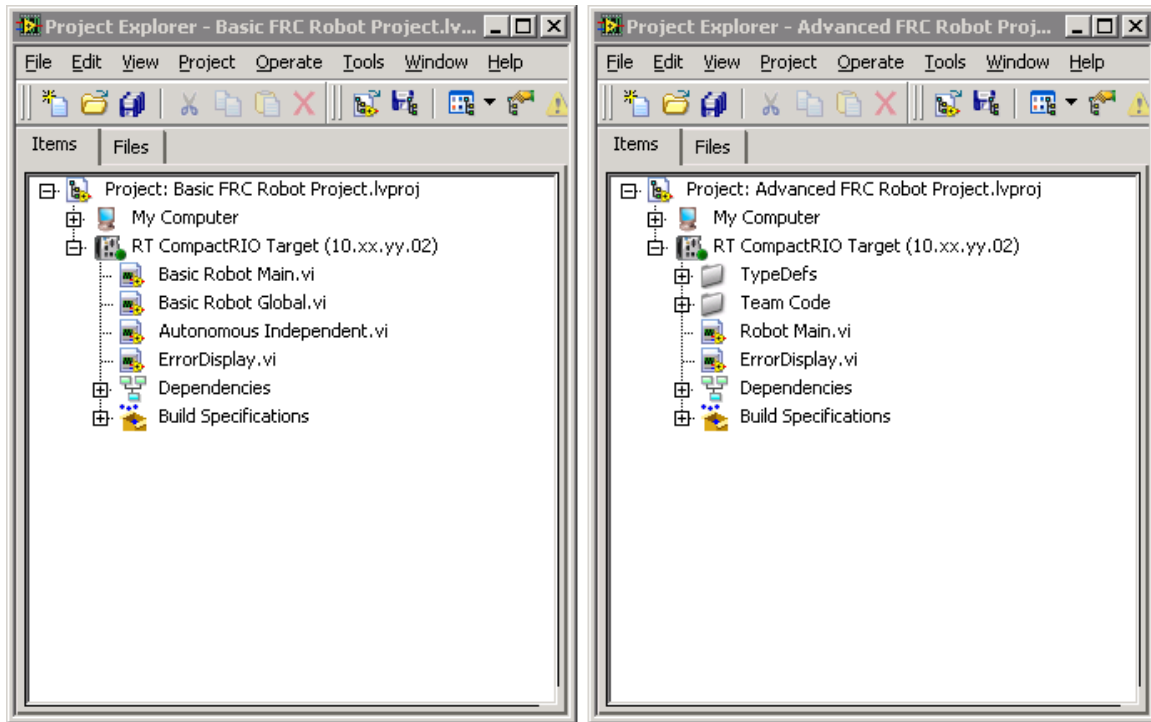


Figure 1: FRC Robot Projects

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

Each FRC robot project also has a Robot Main VI and an ErrorDisplay VI. The Robot Main VI is the master VI for the robot and is the top-level VI for the robotics program you run on the CompactRIO device. The ErrorDisplay VI displays errors and warnings that the WPI Robotics Library VIs generate. You can send these errors to the host computer to view the errors in the FRC dashboard project.

## BASIC FRC ROBOT PROJECT

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

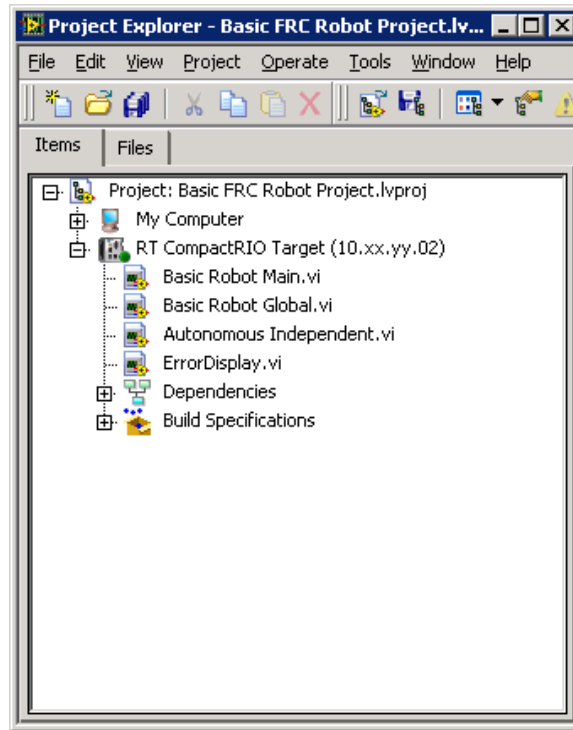


Figure 2: Basic FRC Robot Project

The **RT CompactRIO Target** contains four top-level VIs: Basic Robot Main VI, Basic Robot Global VI, Autonomous Independent VI, and ErrorDisplay VI.

---

## BASIC ROBOT MAIN VI

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

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

---

## COMMUNICATIONS WHILE LOOP

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

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

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

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

---

## IMAGE PROCESSING WHILE LOOP

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

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

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

Before the second While Loop runs, the Camera Open, Set Image Size, and Set Frame Rate VIs configure initial settings for the image acquisition. By default, the Basic Robot Main VI configures an image size of 320x240 pixels and a frame rate of 10 frames per second. The IMAQ Create VI specifies a name of **Primary Image** for any image you retrieve from the acquired image data.

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

---

## PERIODIC TASKS WHILE LOOP

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

---

## BASIC ROBOT GLOBAL VI

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

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

On the block diagram of the Basic Robot Main VI, notice that the **RobotDriveDevRef** output of the Open2Wheel VI is wired to a global variable. The Open2Wheel VI opens a reference to two motors, and the information for these motors is written to the **RobotDriveDevRef** control in the Basic Robot Global VI. If at any time you need to specify a reference to these same motors, you can use the Basic Robot Global VI to provide this data.

---

## AUTONOMOUS INDEPENDENT VI

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

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

---

## PASSING DEVICE REFERENCE DATA WITH THE BASIC ROBOT GLOBAL VI

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

1. Place a Gyro Open VI on the block diagram.
2. Double-click the Gyro Open VI to open the VI.
3. Select the **GyroDevRef** output on the front panel of the Gyro Open VI and press the <Ctrl-C> keys to copy the gyroscope device reference.
4. In the **Project Explorer** window of the basic FRC robot project, double-click the **Basic Robot Global.vi** item to open the Basic Robot Global VI.
5. Press the <Ctrl-V> keys to paste the **GyroDevRef** control on the front panel of the Basic Robot Global VI.
6. Save the Basic Robot Global VI.

7. Drag the VI icon of the Basic Robot Global VI to the block diagram of the VI in which you placed the Gyro Open VI. Notice that LabVIEW places a global variable on the block diagram. By default, the global variable is associated with the first front panel object with an owned label you added to the global VI. In this case, the global variable is associated with the **RobotDriveDevRef** control.
8. Right-click the **RobotDriveDevRef** global variable and select **GyroDevRef** from the **Select Item** shortcut menu to associate the global variable with the data from the **GyroDevRef** control.
9. Wire the **GyroDevRef** output of the Gyro Open VI to the **GyroDevRef** global variable.
10. When you configure the other parameters of the Gyro Open VI and run the VI, the Gyro Open VI opens a reference to a gyroscope, and LabVIEW writes the gyroscope device reference data to the **GyroDevRef** control of the Basic Robot Global VI.

---

## ERRORDISPLAY VI

The ErrorDisplay VI displays errors and warnings that the WPI Robotics Library VIs generate. You can specify whether this VI displays an error dialog each time an error occurs. You also can specify whether to generate an error log and whether to send the error information to the host computer.

## ADVANCED FRC ROBOT PROJECT

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

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

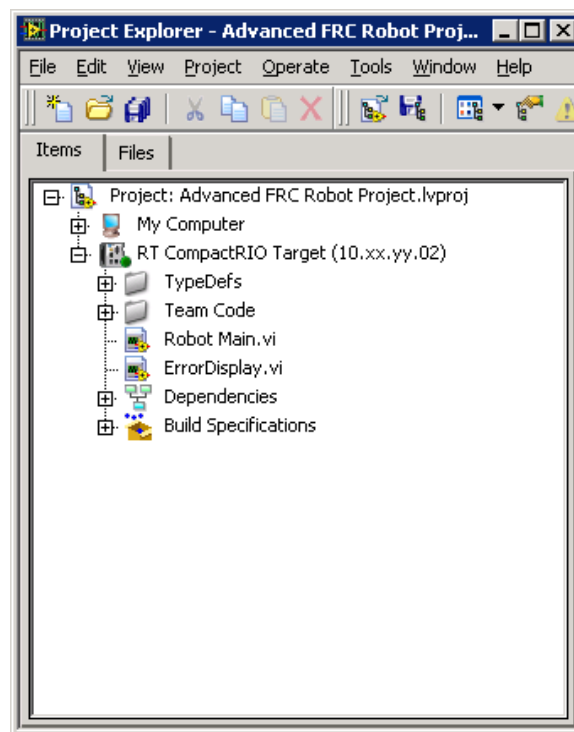


Figure 3: Advanced FRC Robot Project

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

---

## ROBOT MAIN VI

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

In the **Project Explorer** window, double-click the **Robot Main.vi** item to open the Robot Main VI. Select **Window»Show Block Diagram** or press the <Ctrl-E> keys to view the block diagram. The block diagram contains a single While Loop and a number of subVIs. The While Loop determines the behavior of the robot according to the competition mode and state. When the competition state is Autonomous Enabled, the Robot Main VI calls the Autonomous Iterative VI. When the competition state is TeleOp Enabled, the Robot Main VI calls the TeleOp VI. If the robot is in Disabled status, the Robot Main VI calls the Disable 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 Robot Communication VI and the Send Dashboard Data VI to establish communication with the driver station and send data regularly to the FRC dashboard project on the host computer. The Robot Main VI runs until the **Abort Execution** button on the VI toolbar is pressed or until the **Finish** button on the front panel is pressed. If the **Abort Execution** button is pressed, the Robot Main VI aborts immediately and does not perform any cleanup tasks. If the **Finish** button is pressed, the Robot Main VI calls the Finish VI, which you can configure to close device references, save collected data to file, or perform any other cleanup tasks.

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

---

## ERRORDISPLAY VI

The ErrorDisplay VI displays errors and warnings that the WPI Robotics Library VIs generate. You can specify whether this VI displays an error dialog each time an error occurs. You also can specify whether to generate an error log and whether to send the error information to the host computer.

---

## TEAM CODE VIS

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

---

## BEGIN VI

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

By default, the Begin VI configures settings for the Autonomous program you want to run, opens a reference to the Axis camera, configures options for the FRC dashboard project on the host computer, and defines the RobotData

and `PeriodicTaskData` type definitions. You can use the WPI Robotics Library VIs and other LabVIEW VIs to configure other initialization tasks.

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

---

## AUTONOMOUS INDEPENDENT VI

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

Notice that the default block diagram of the Autonomous Independent VI is empty. You can use the WPI Robotics Library VIs or other LabVIEW VIs to specify the program you want to run in Autonomous mode.

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

---

## AUTONOMOUS ITERATIVE VI

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

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

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

---

## DISABLED VI

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

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

---

## TELEOP VI

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

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

The TeleOp VI uses the RobotData type definition to specify device references of the correct data types for use with the WPI Robotics Library VIs. Refer to the [RobotData Type Definition](#) section of this document for information about the RobotData type definition.

---

## ROBOT GLOBAL DATA VI

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

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

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

---

## VISION PROCESSING VI

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

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

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

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

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

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

---

## PERIODIC TASKS VI

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

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

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

---

## FINISH VI

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

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

---

## TYPE DEFINITIONS

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

---

## ROBOTDATA TYPE DEFINITION

The RobotData type definition specifies data types and values for the device references you use with the WPI Robotics Library VIs. You can wire elements of this cluster to the appropriate parameters of the WPI Robotics Library VIs. For example, you can wire the **JoystickDevRef** output of the Joystick Open VI to this type definition. The type definition maintains the values you specified in the Open VI. If you then wire this type definition to the **JoystickDevRef** input of the Joystick Get VI, the Joystick Get VI uses the persisted values of the type definition to determine the joystick for which to return button and axis information.

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

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

---

## VISIONDATA TYPE DEFINITION

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

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

---

## PERIODICTASKDATA TYPE DEFINITION

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

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

## DEPLOYING THE FRC cRIO ROBOT PROJECT

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

---

### DEPLOYING THE PROGRAM USING THE RUN BUTTON

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

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

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

---

### DEPLOYING THE PROGRAM FROM THE PROJECT EXPLORER WINDOW

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

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

---

### BUILDING AND DEPLOYING A STAND-ALONE APPLICATION

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

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

1. In the **Project Explorer** window, right-click **Build Specifications** under the **RT CompactRIO Target** and select **New»Real-Time Application** from the shortcut menu to display the **Real-Time Application Properties** dialog box.
2. On the **Information** page, specify a name for the build specification in the **Build specification name** text box.
3. Specify a name for the application in the **Target filename** text box.
4. Specify the location on the host computer to which you want to save the stand-alone application in the **Local destination directory** field.
5. Specify the location on the CompactRIO device to which you want to save the stand-alone application in the **Target destination directory** field.
6. Select **Source Files** from the **Category** list.
7. From the **Project Files** tree on the **Source Files** page, select **Robot Main.vi**.
8. Click the **Add Item** arrow button next to the **Startup VIs** list to move the Robot Main VI to the **Startup VIs** list.
9. Select **Additional Exclusions** from the **Category** list.
10. On the **Additional Exclusions** page, remove the checkmark from the **Modify project library file after removing unused members** checkbox.
11. Click the **OK** button to close the **Real-Time Application Properties** dialog box.
12. In the **Project Explorer** window, right-click the build specification you created under the **RT CompactRIO Target** and select **Build** from the shortcut menu to build the application.
13. Right-click the build specification and select **Run as startup** from the shortcut menu to set the application as the startup application and deploy the application to the CompactRIO device. LabVIEW prompts you to reboot the RT target.
14. Reboot the CompactRIO device to run the application.

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

---

## CONNECTING TO THE COMPACTRIO DEVICE

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

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

## FRC DASHBOARD PROJECT

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

Complete the following steps to create an FRC dashboard project.

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

The FRC dashboard project looks similar to the following figure.

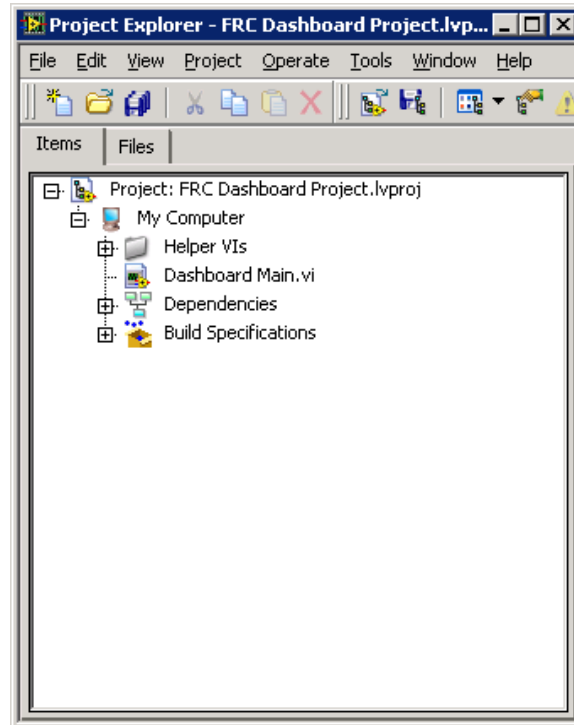


Figure 4: FRC Dashboard Project

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

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

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

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

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

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

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

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

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

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

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

## COPYRIGHT

© 2008 National Instruments Corporation. All rights reserved.

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

## TRADEMARKS

National Instruments, NI, [ni.com](http://ni.com), and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on [ni.com/legal](http://ni.com/legal) for more information about National Instruments trademarks.

## PATENTS

For patents covering the National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or [ni.com/patents](http://ni.com/patents).