

FRC LABVIEW PROGRAMMING

FRC LabVIEW Programming

Table of Contents

Setting up the Development Environment	3
Installing LabVIEW for FRC 2019 (LabVIEW only)	4
Installing the FRC Update Suite (All Languages).....	27
Creating Robot Programs	48
Creating, Building and Loading your Benchtop Test Program.....	49
Tank Drive Tutorial	55
Command and Control Tutorial	60
LabVIEW Resources	82
LabVIEW Resources	83
Waiting for Target to Respond - Recovering from bad loops.....	84
Talon SRX CAN.....	88
How To Toggle Between Two Camera Modes.....	89
LabVIEW Examples and Tutorials.....	90
Add an Independent Motor to a Project	92
Keyboard Navigation with the roboRIO	94
Making a One-Shot Button Press.....	96
Adding Safety Features to Your Robot Code	102
How to Use Joystick Buttons to Control Motors or Solenoids.....	104
Local and Global Variables in LabVIEW for FRC	108
Using the Compressor in LabVIEW	112

Setting up the Development Environment

Installing LabVIEW for FRC 2019 (LabVIEW only)

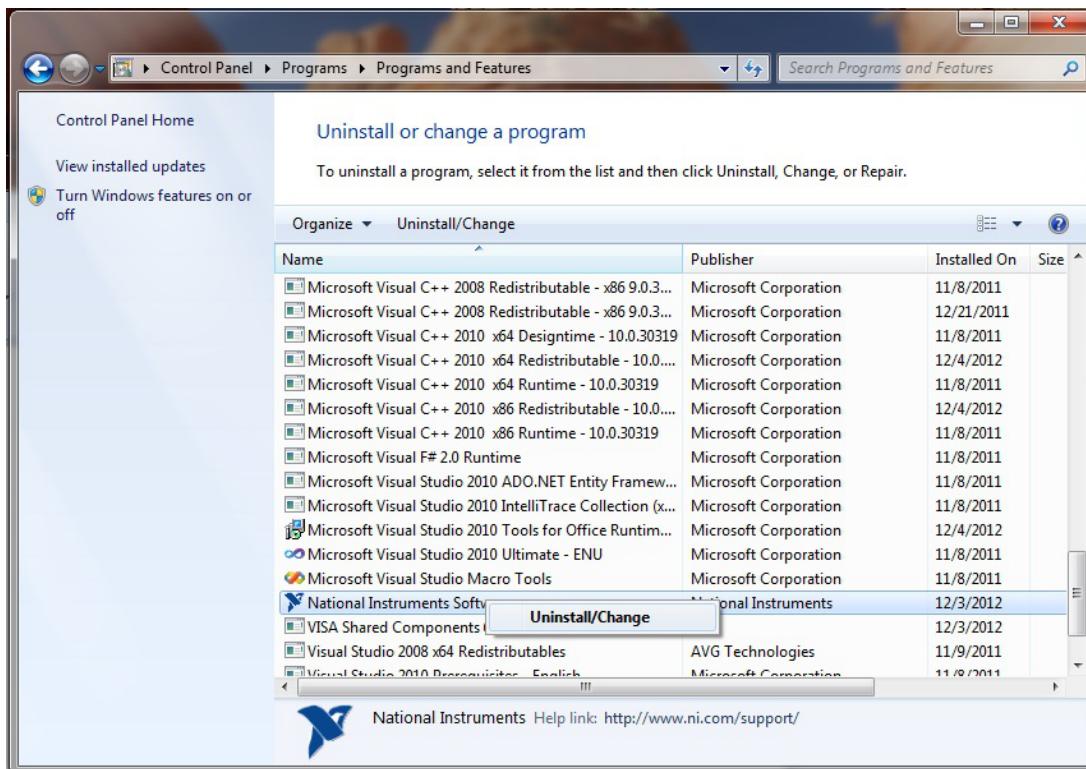


Note: This installation is for teams programming in LabVIEW or using NI Vision Assistant only. C++ and Java teams not using these features do not need to install from the DVD.

Download and installation times will vary widely with computer and internet connection specifications, however note that this process involves a large file download and installation and will likely take at least an hour to complete.

FRC LabVIEW Programming

Uninstall Old Versions (Recommended)

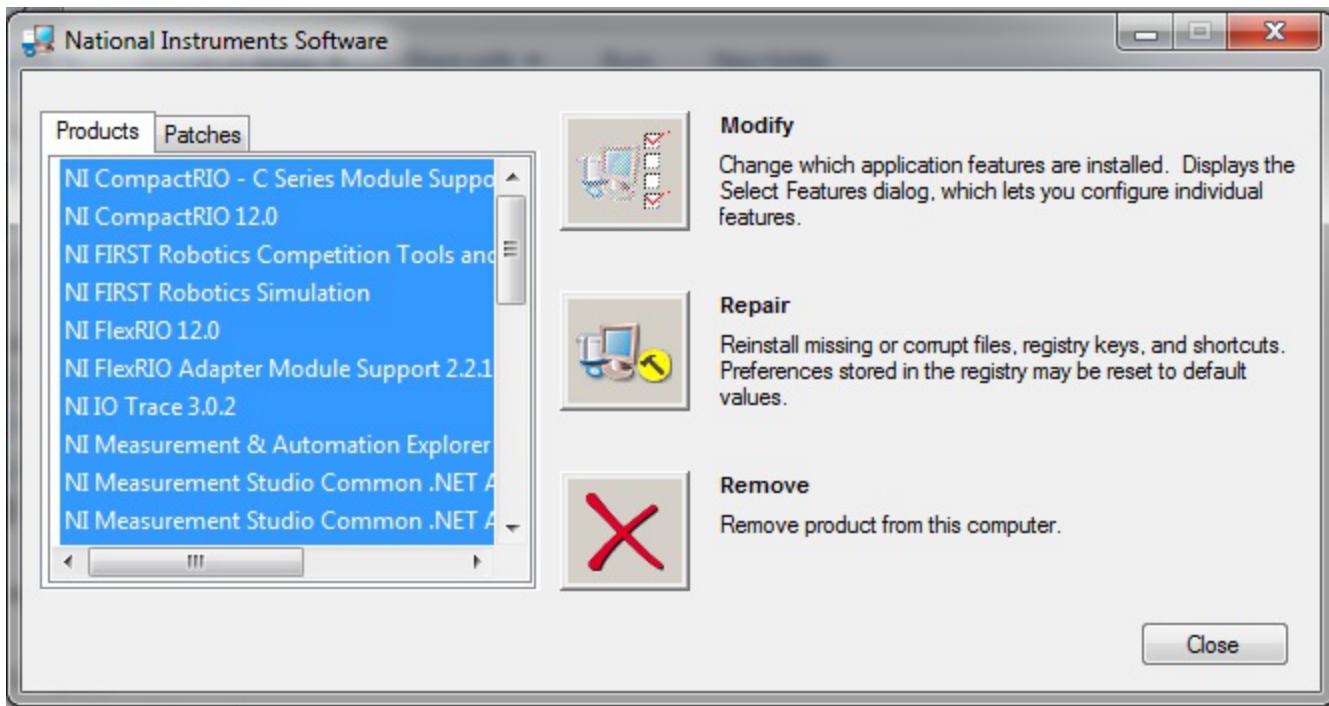


NOTE: If you wish to keep programming cRIOs you will need to maintain an install of LabVIEW for FRC 2014. The LabVIEW for FRC 2014 license has been extended. While these versions should be able to co-exist on a single computer, this is not a configuration that has been extensively tested.

Before installing the new version of LabVIEW it is recommended to remove any old versions. The new version will likely co-exist with the old version, but all testing has been done with FRC 2019 only. Make sure to back up any team code located in the "User\LabVIEW Data" directory before un-installing. Then click Start >> Control Panel >> Uninstall a Program. Locate the entry labeled "National Instruments Software", right-click on it and select Uninstall/Change.

FRC LabVIEW Programming

Select Components to Uninstall



In the left pane of the dialog box that appears, select all entries. The easiest way to do this is to click the top entry to highlight it, then scroll down to the bottom entry, press and hold shift and click on the last entry then release shift. Click Remove. Wait for the uninstaller to complete and reboot if prompted.

Getting LabVIEW Installer

Either locate and insert the LabVIEW USB Drive or download the LabVIEW 2019 installer from <http://www.ni.com/download/labview-for-frc-18.0/7841/en/>

If downloaded, right click on the downloaded file (NI_FRC2019.zip) and select Extract All.

Note: This is a large download (~5GB). It is recommended to use a fast internet connection and to use the NI Downloader to allow the download to resume if interrupted.

FRC LabVIEW Programming

Installing LabVIEW

National Instruments LabVIEW requires a license. Each season's license is active until January 31st of the following year (e.g. the license for the 2019 season expires on January 31, 2020)

Teams are permitted to install the software on as many team computers as needed, subject to the restrictions and license terms that accompany the applicable software, and provided that only team members or mentors use the software, and solely for FRC. Rights to use LabVIEW are governed solely by the terms of the license agreements that are shown during the installation of the applicable software.

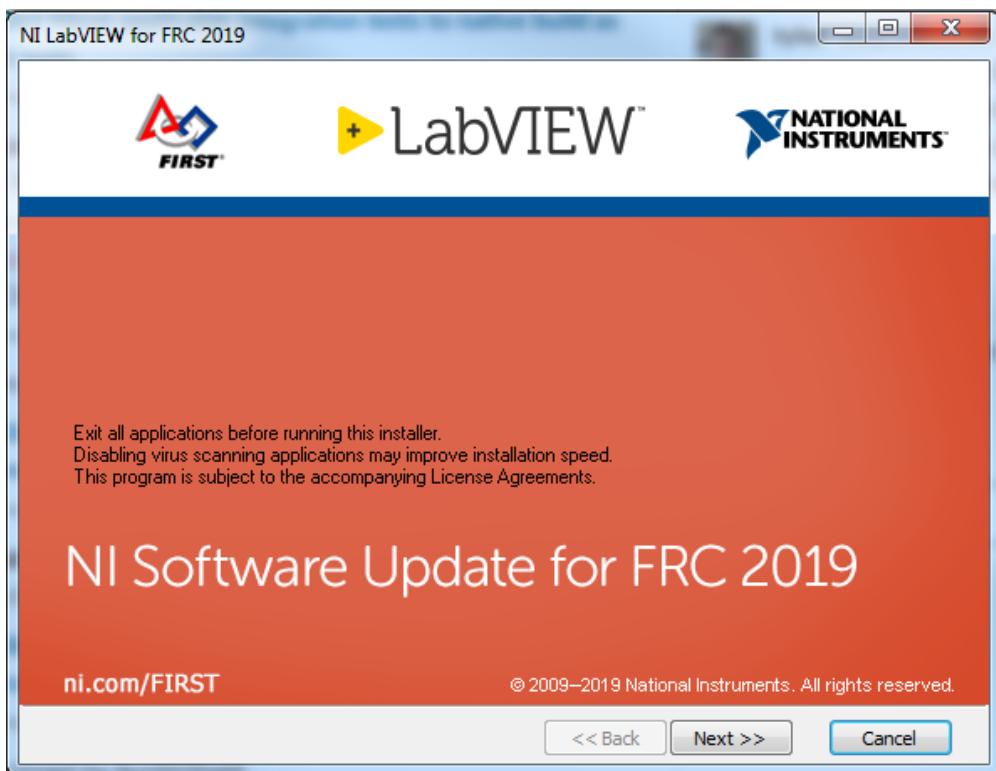
Welcome



Double click on autorun.exe to launch the installer. If prompted to allow changes click Yes. To install LabVIEW to program your FRC robot, click the top option **Install Everything for LabVIEW Development**. To install only NI Vision Assistant for use with C++ or Java, click **Install Only NI Vision Development Module**. If prompted with any Windows security warnings, click Allow or Yes.

FRC LabVIEW Programming

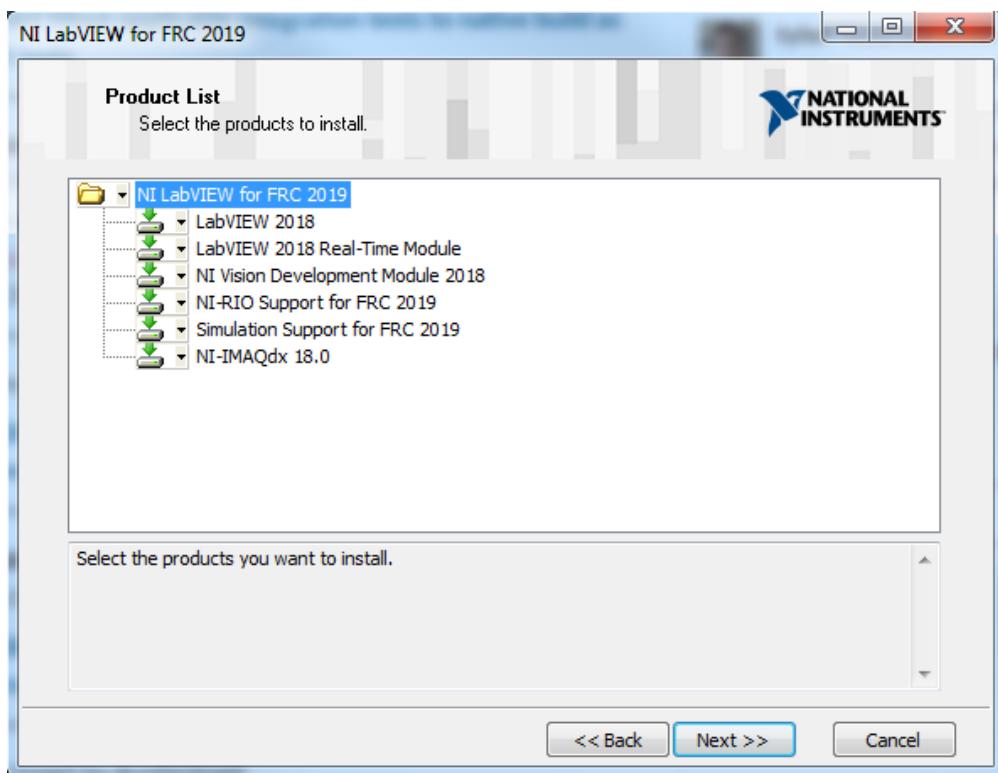
Warnings



Click "Next"

FRC LabVIEW Programming

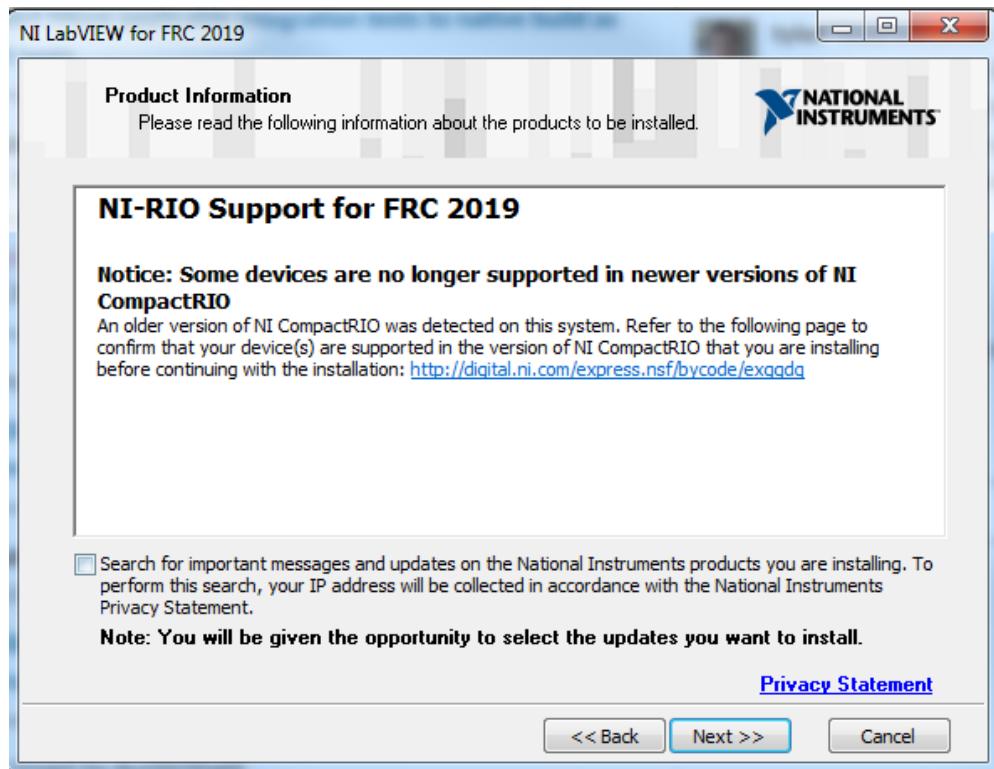
Product List



Click "Next"

FRC LabVIEW Programming

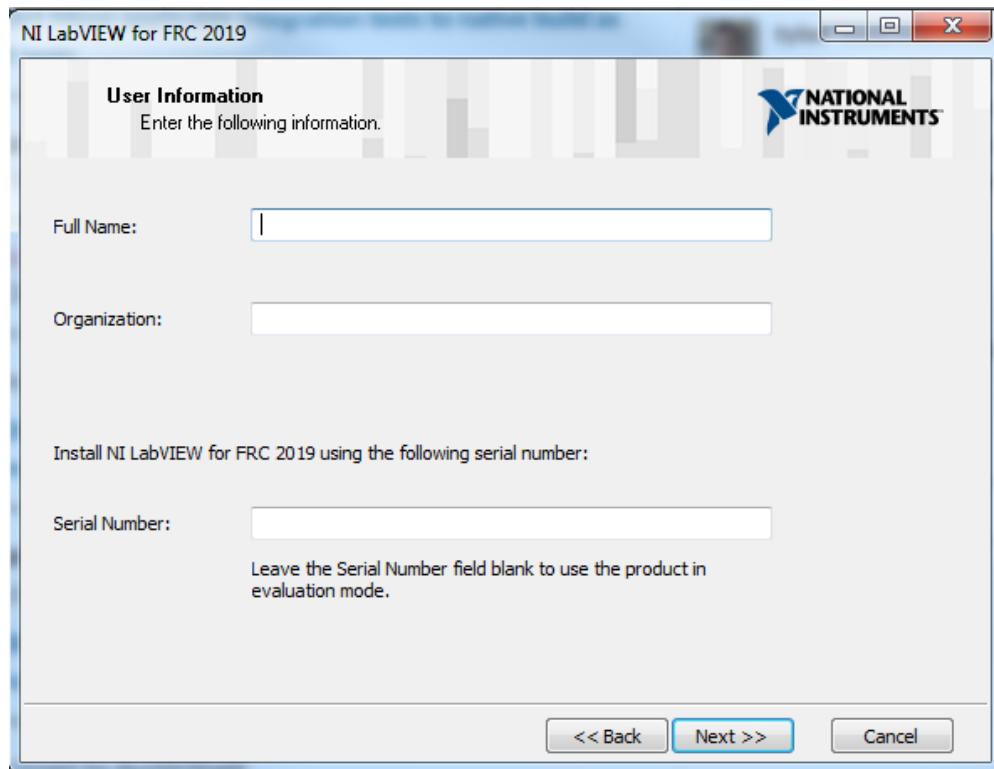
Product Information



Un-check the box, then click "Next". (Note: you may not see the warning shown in the top portion of the window in this picture).

FRC LabVIEW Programming

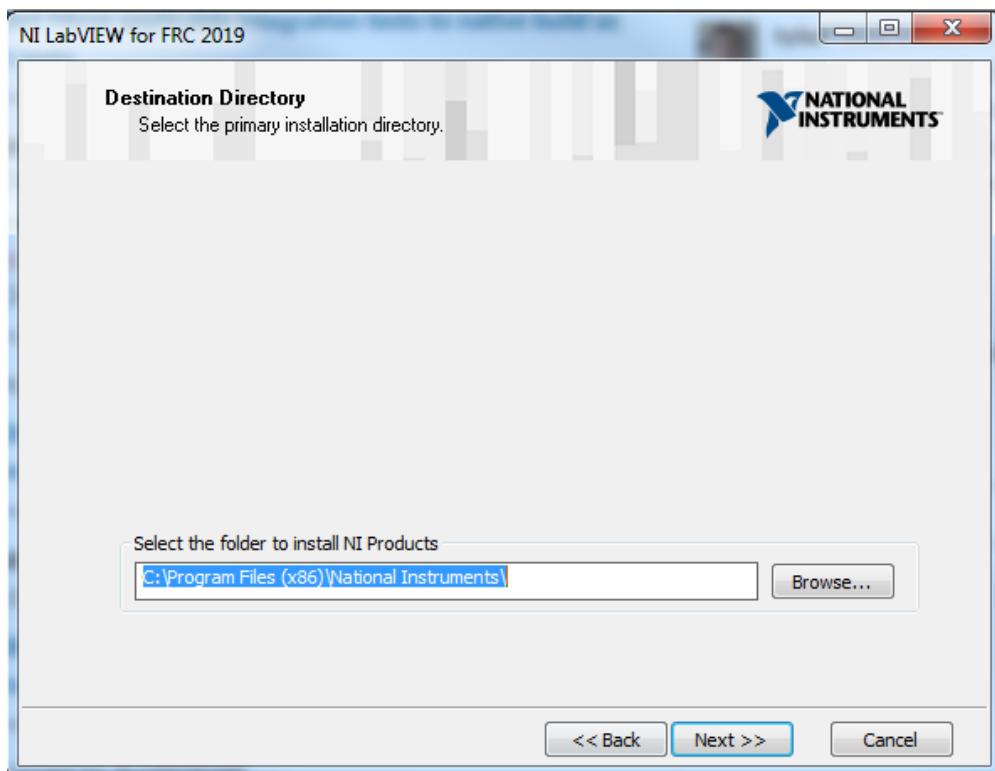
User Information



Enter name, organization, and the serial number from the [ReadMe in the File Releases on Teamforge](#). Click "Next". If you cannot find your serial number, please reach out to National Instruments at www.ni.com/frc/needhelp.

FRC LabVIEW Programming

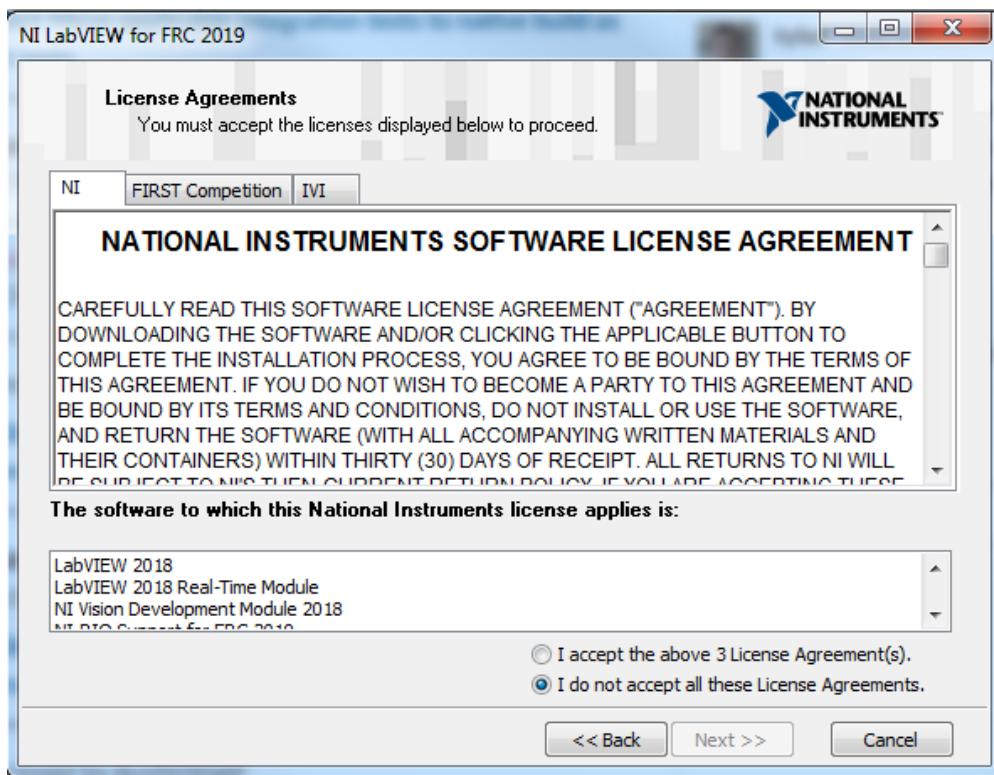
Destination Directory



Click "Next"

FRC LabVIEW Programming

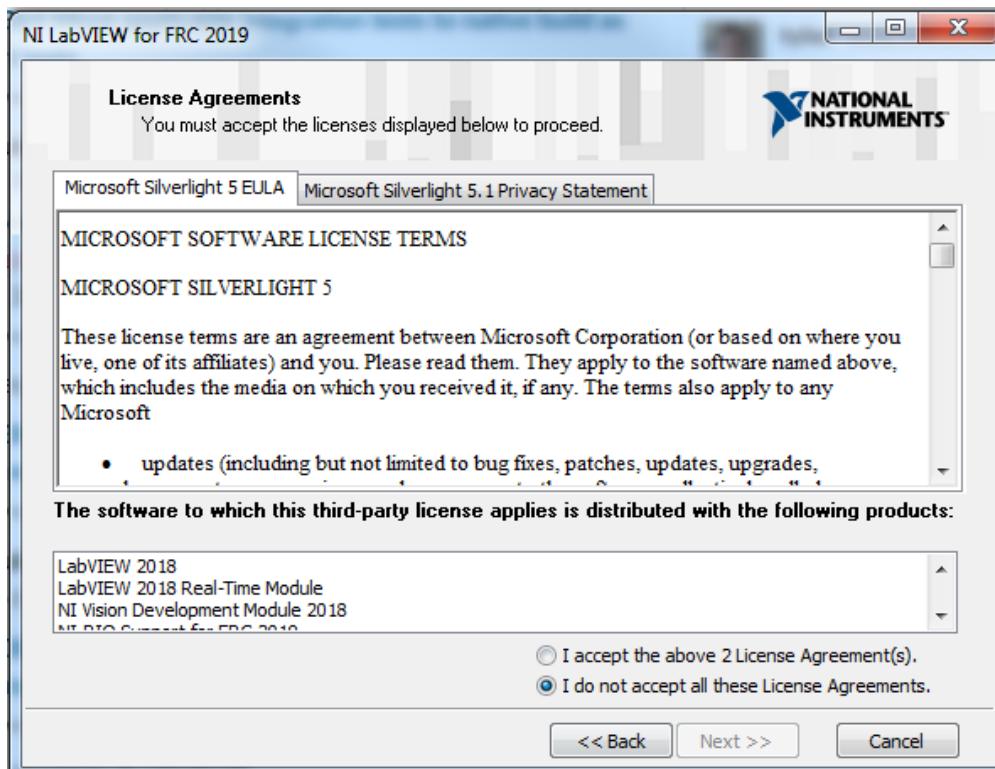
License Agreements (1)



Check "I accept..." then Click "Next"

FRC LabVIEW Programming

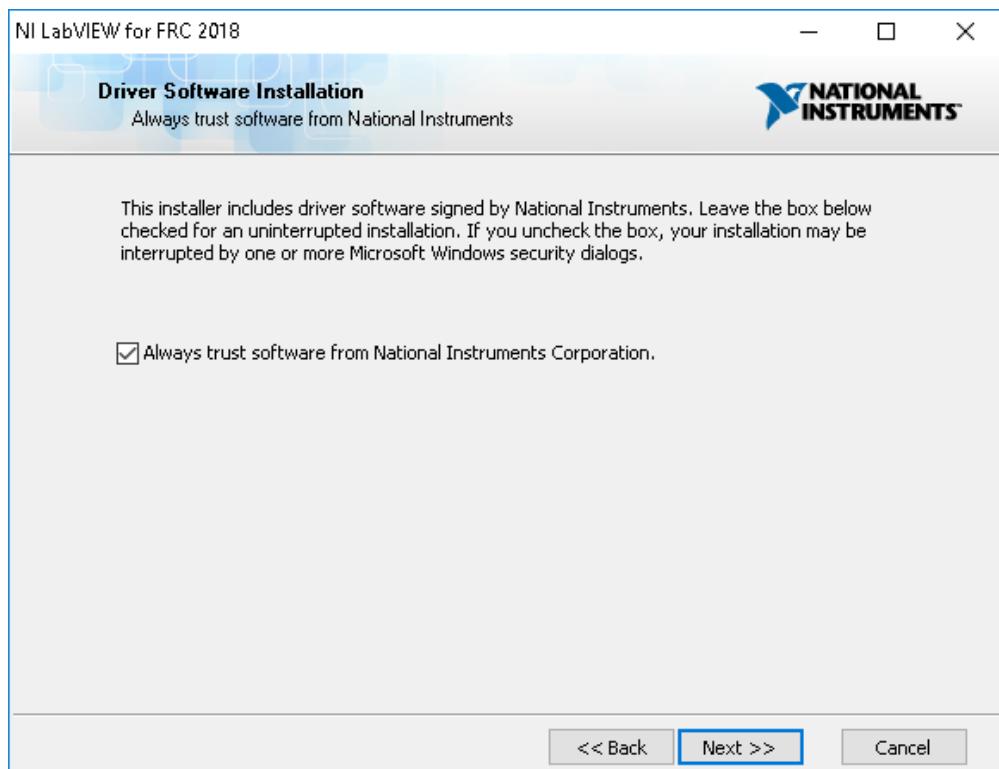
License Agreements (2)



Check "I accept..." then Click "Next"

FRC LabVIEW Programming

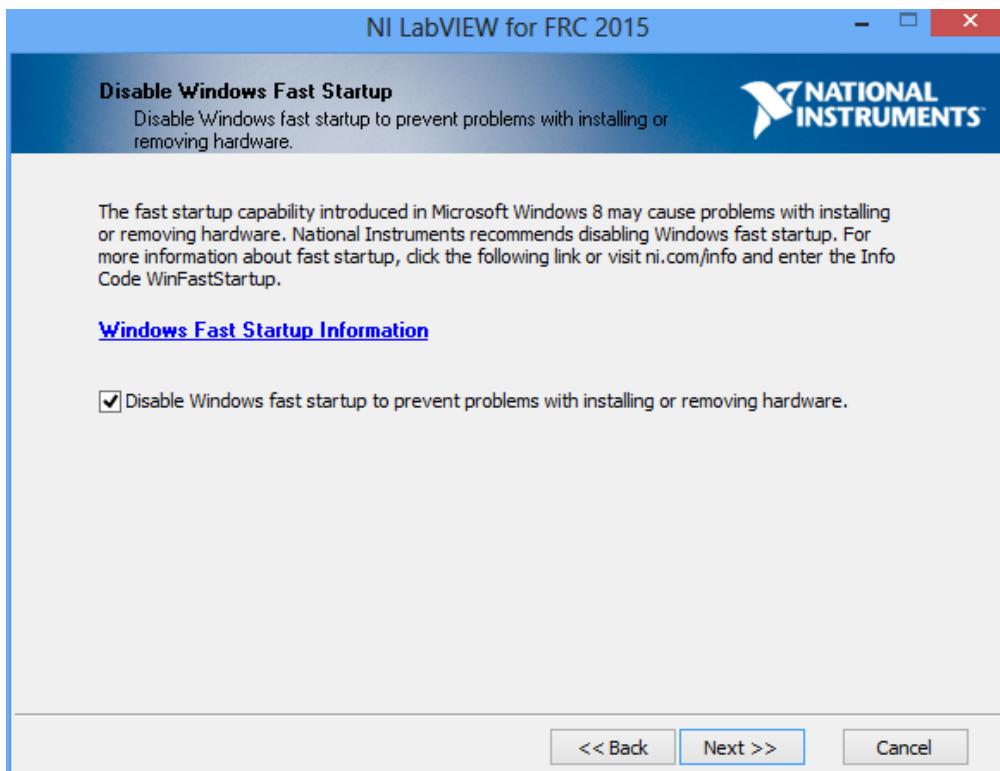
Driver Software Installation



If you see this screen, Click "Next"

FRC LabVIEW Programming

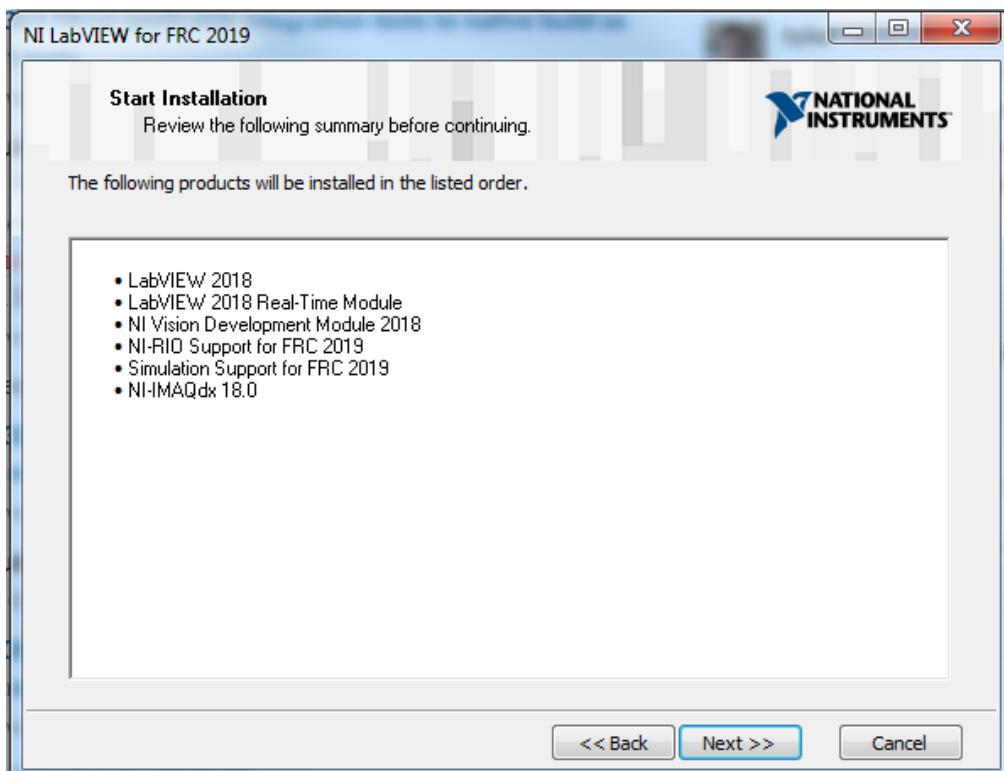
Disable Windows Fast Startup



If you see this screen, click "Next"

FRC LabVIEW Programming

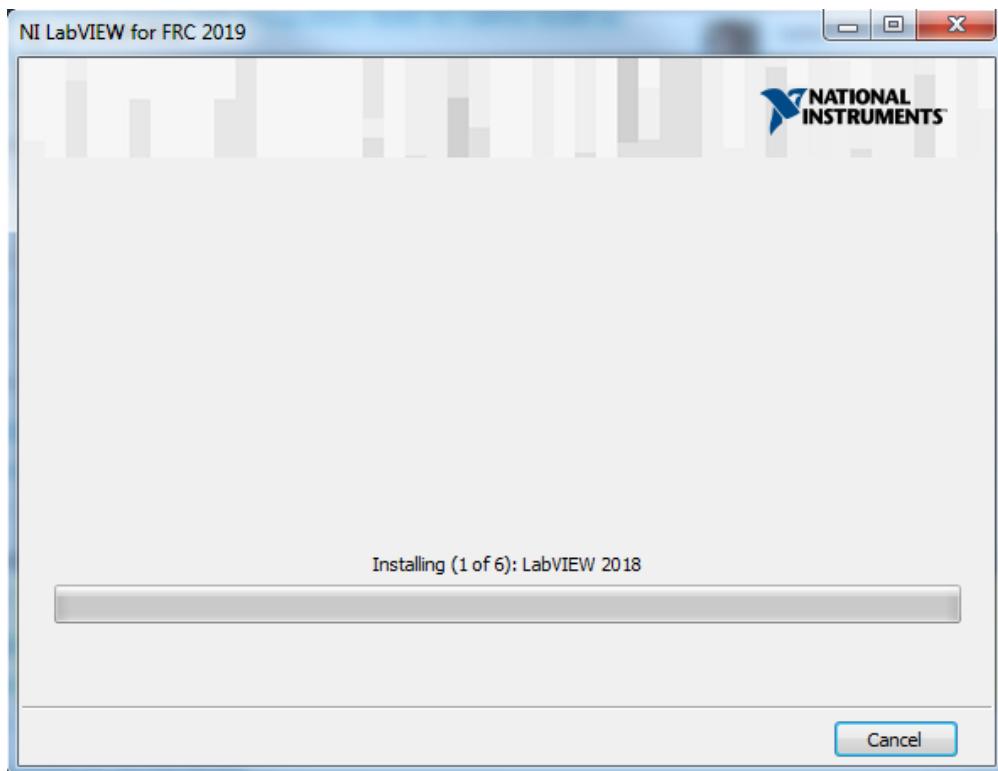
Start Installation



Click "Next"

FRC LabVIEW Programming

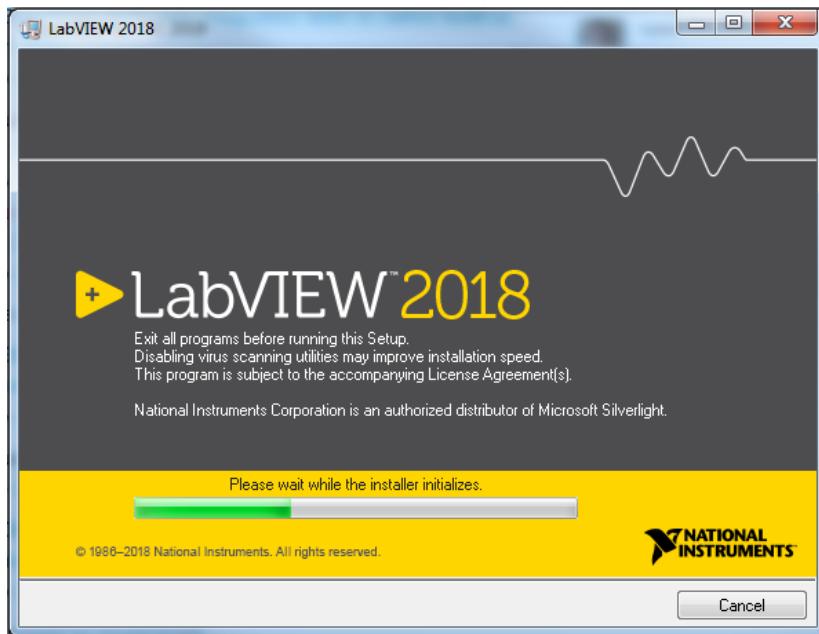
Overall Progress



Overall installation progress will be tracked in this window

FRC LabVIEW Programming

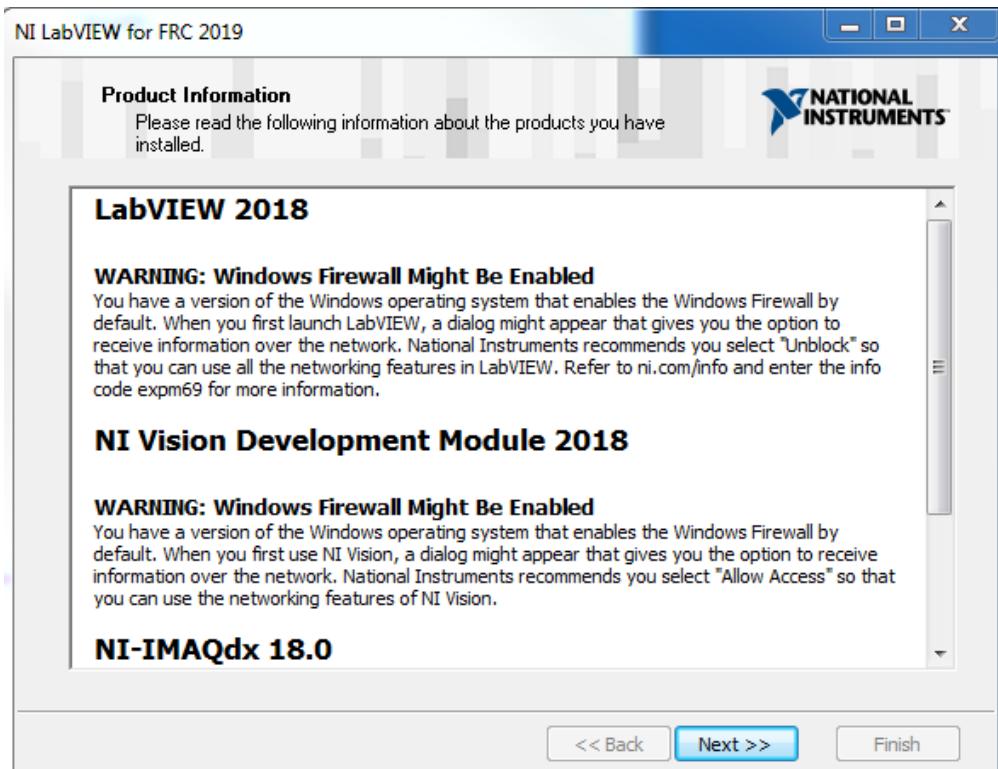
Individual Product Progress



Each product installed will also create an individual progress window like the one shown above.

FRC LabVIEW Programming

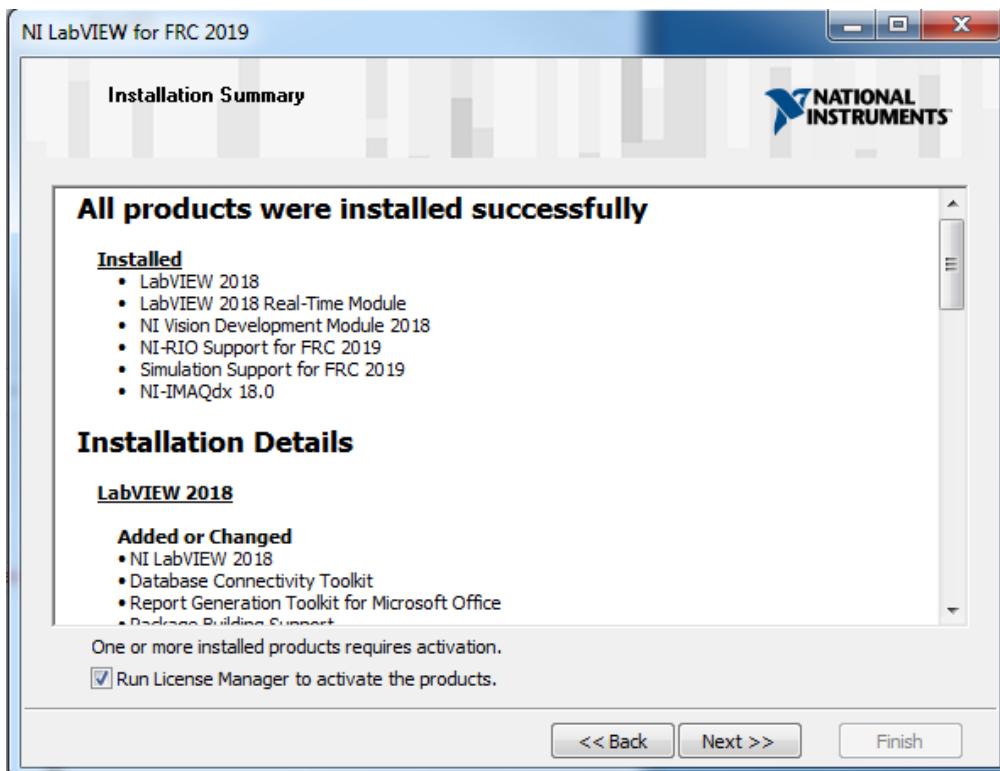
Product Information



Click "Next"

FRC LabVIEW Programming

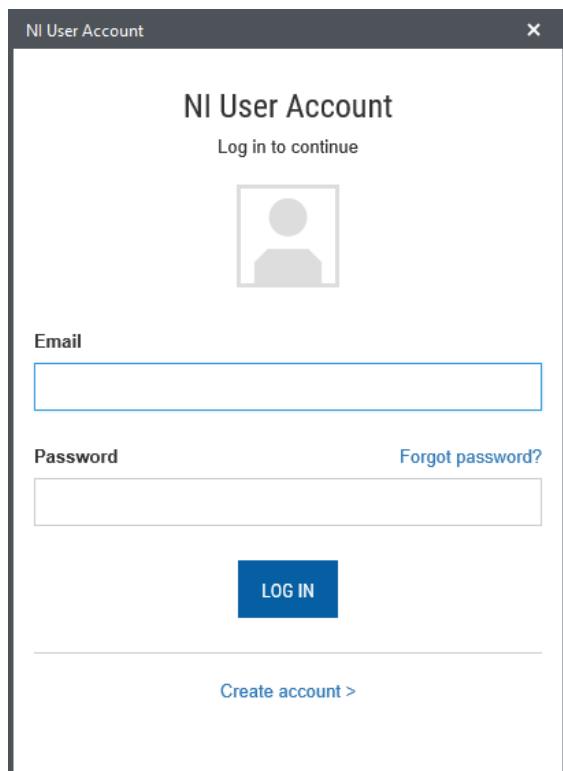
Installation Summary



If internet access is available and you are ready to activate, click "Next"; otherwise uncheck the "Run License Manager..." and click "Next".

FRC LabVIEW Programming

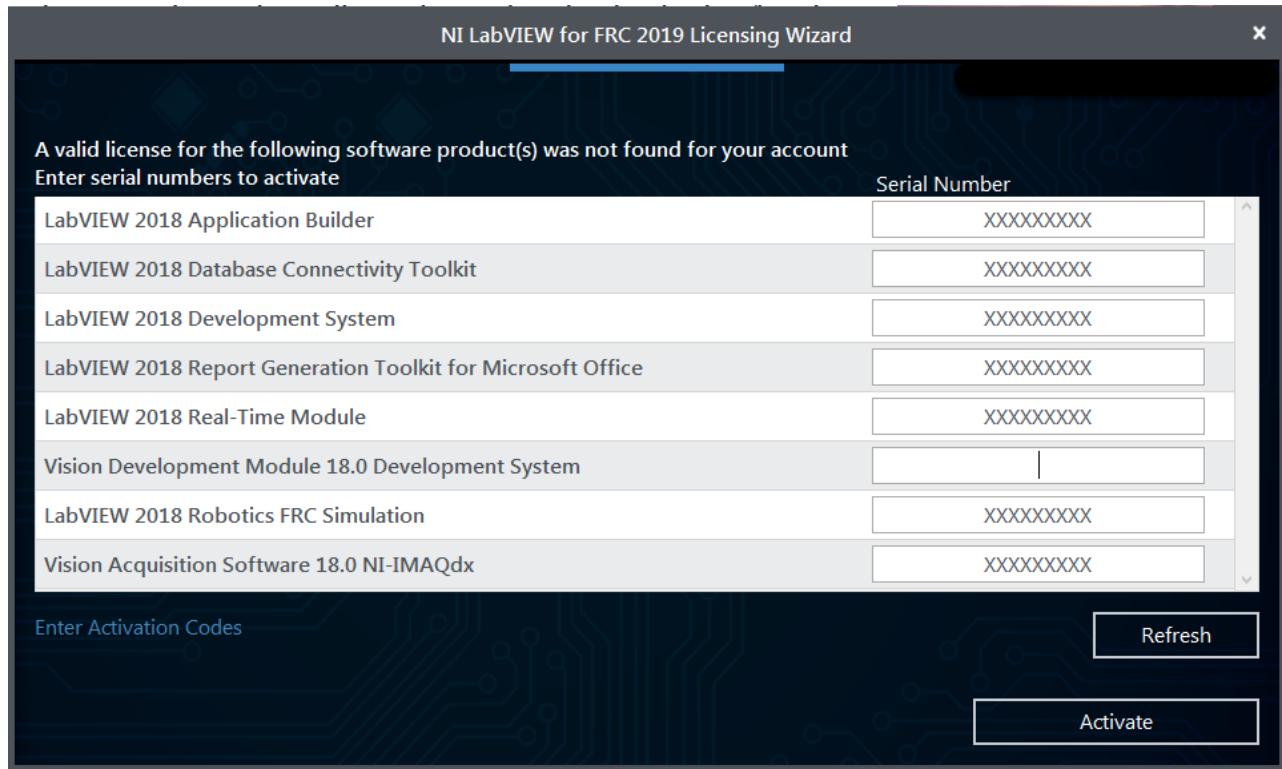
NI Activation Wizard



Log into your ni.com account. If you don't have an account, select 'Create account' to create a free account.

FRC LabVIEW Programming

NI Activation Wizard (2)

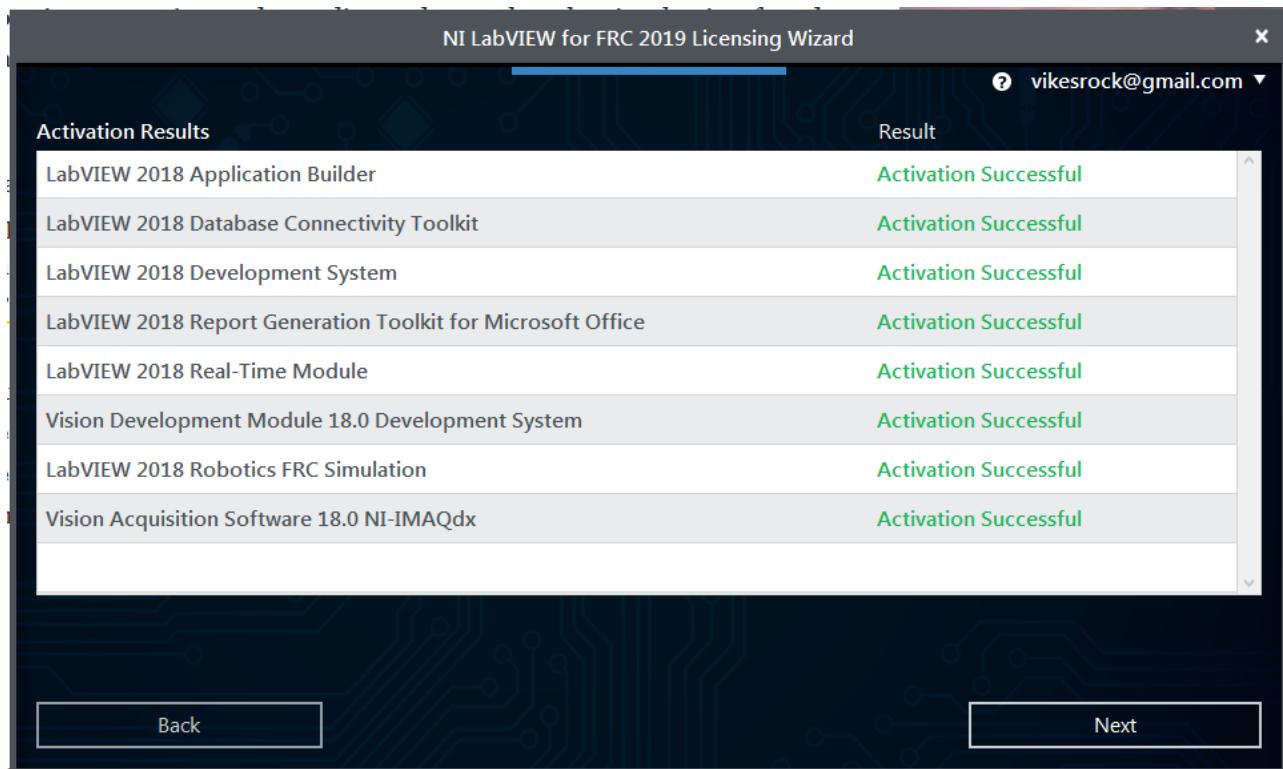


The serial number you entered at the "User Information" screen should appear in all of the text boxes, if it doesn't, enter it now. Click "Activate".

Note: If this is the first time activating the 2019 software on this account, you will see the message shown above about a valid license not being found. You can ignore this.

FRC LabVIEW Programming

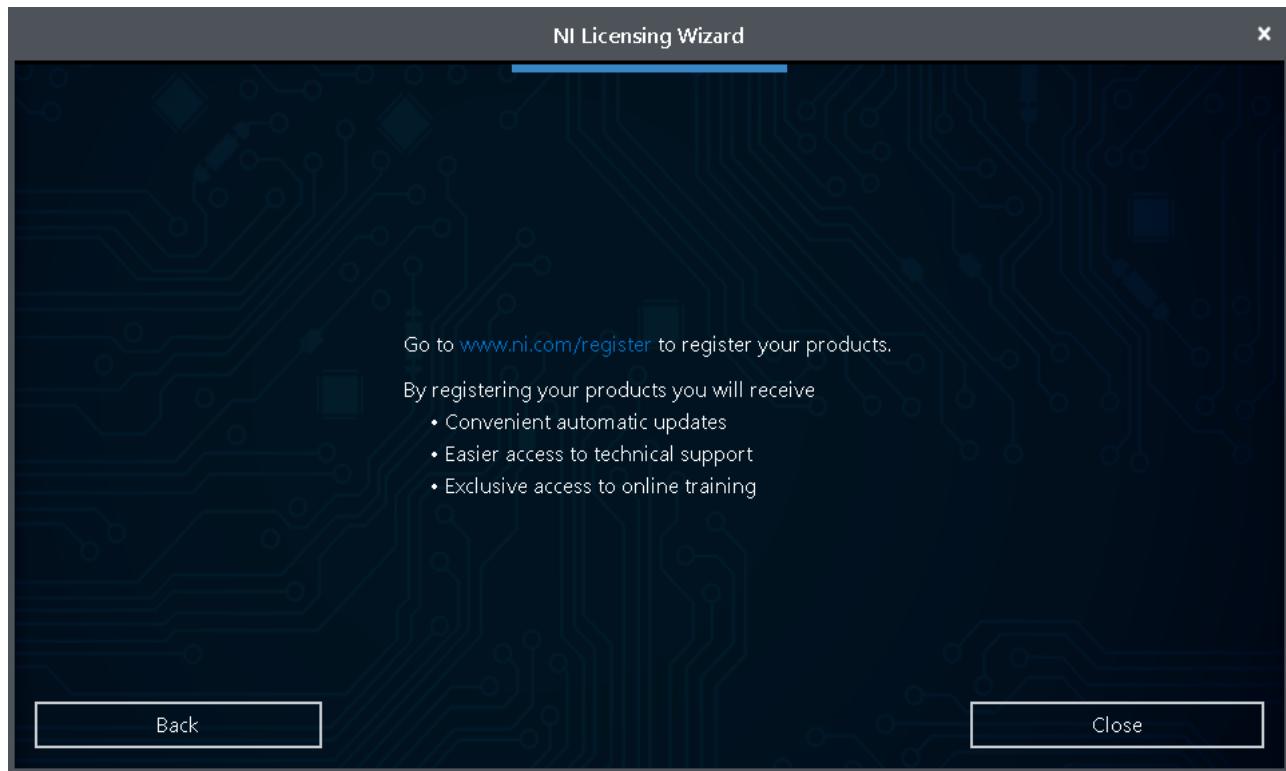
NI Activation Wizard (3)



If your products activate successfully, an "Activation Successful" message will appear. If the serial number was incorrect, it will give you a text box and you can re-enter the number and select "Try Again". If everything activated successfully, click "Next".

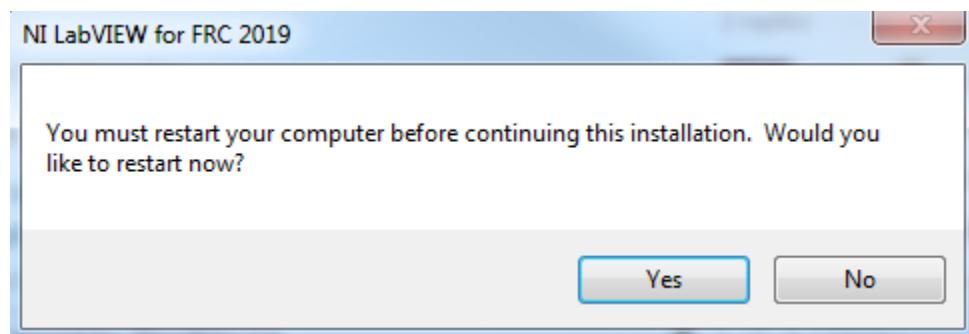
FRC LabVIEW Programming

NI Activation Wizard (4)



Click "Close".

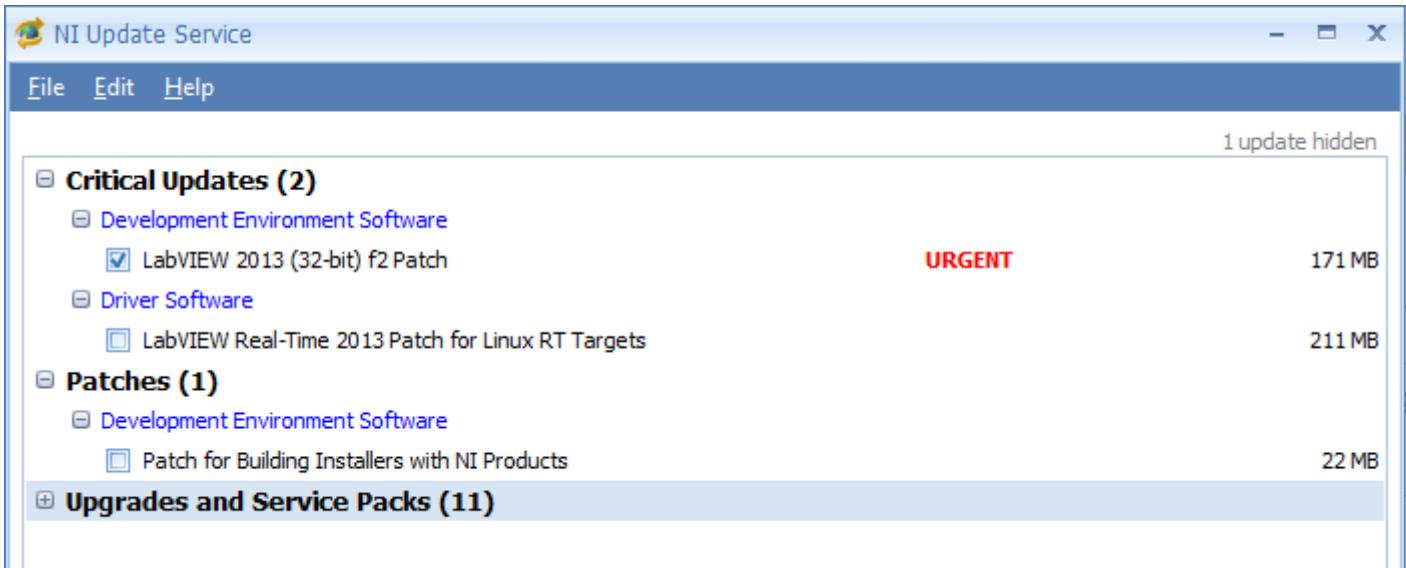
Restart Message



Select "Yes"

FRC LabVIEW Programming

NI Update Service



On occasion you may see alerts from the NI Update Service about patches to LabVIEW. **It is not recommended to install these updates unless directed by FRC through our usual communication channels (Frank's Blog, Team Updates or E-mail Blasts).**

Installing the FRC Update Suite (All Languages)



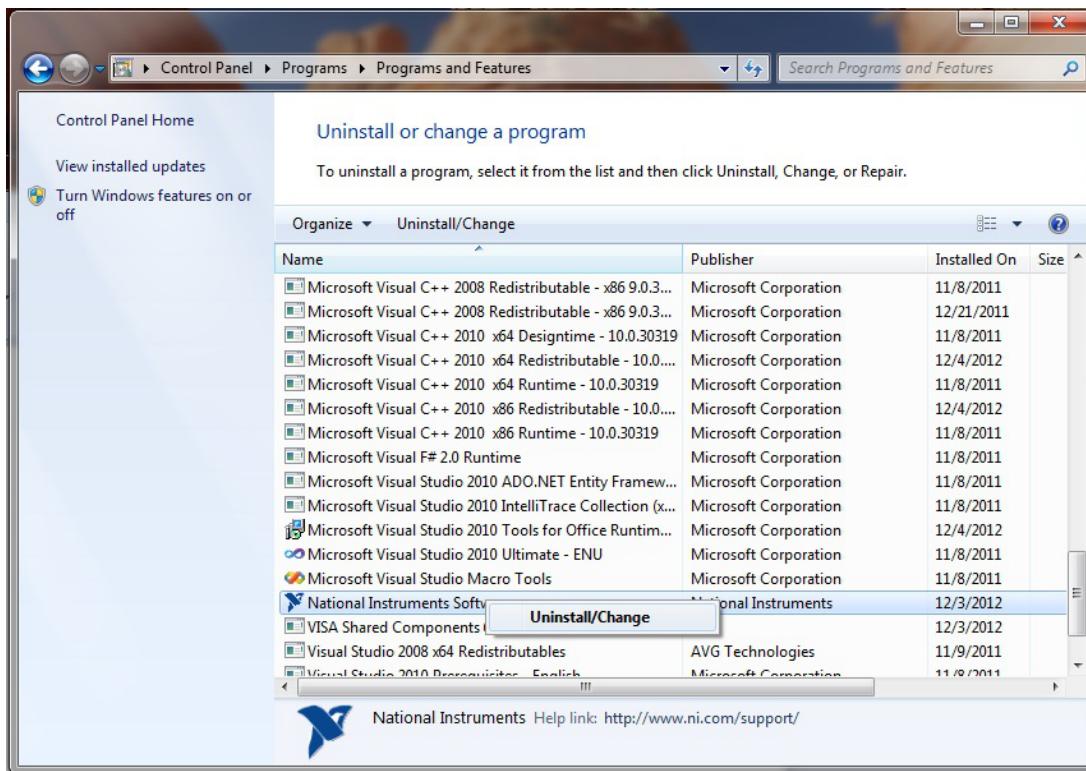
The FRC Update Suite contains the following software components: LabVIEW Update, FRC Driver Station, and FRC Utilities. If an FRC LabVIEW installation is found, the LabVIEW Update will be installed or updated, otherwise this step will be skipped. The FRC Driver Station and FRC Utilities will always be installed or updated. The LabVIEW runtime components required for the driver station and utilities is included in this package. No components from the LabVIEW Merged Suite are required for running either the Driver Station or Utilities.

C++ and Java teams wishing to use NI Vision Assistant should run the full Suite installer as described in the article - [Installing LabVIEW for FRC \(LabVIEW only\)](#)

! Note: The Driver Station will only work on Windows 7, Windows 8, Windows 8.1, and Windows 10. It will not work on Windows XP.

FRC LabVIEW Programming

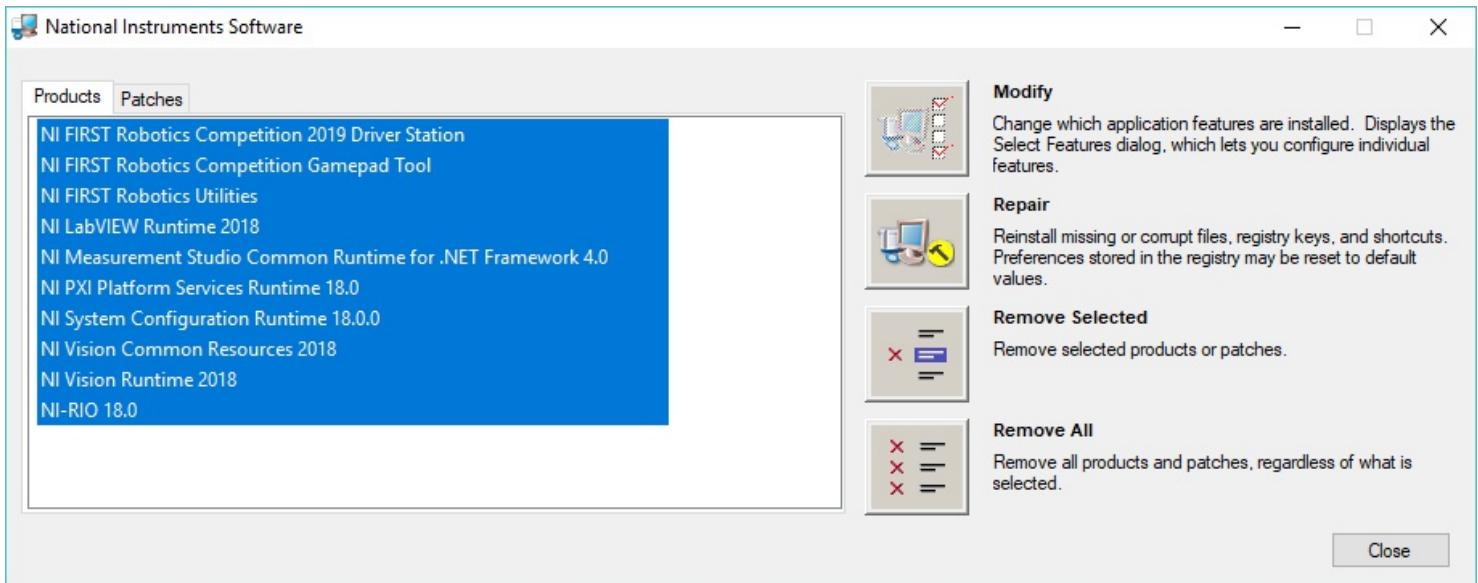
Uninstall Old Versions (Recommended)



LabVIEW teams have already completed this step, do not repeat it. Before installing the new version of the NI Update it is recommended to remove any old versions. The new version will likely properly overwrite the old version, but all testing has been done with FRC 2019 only. Make sure to back up any team code located in the "User\LabVIEW Data" directory before un-installing. Then click Start >> Control Panel >> Uninstall a Program. Locate the entry labeled "National Instruments Software", right-click on it and select Uninstall/Change.

FRC LabVIEW Programming

Select Components to Uninstall



Click Remove All and follow any prompts to remove all previous NI products.

Downloading the Update

Download the update from <http://www.ni.com/download/first-robotics-software-2017/7904/en/>

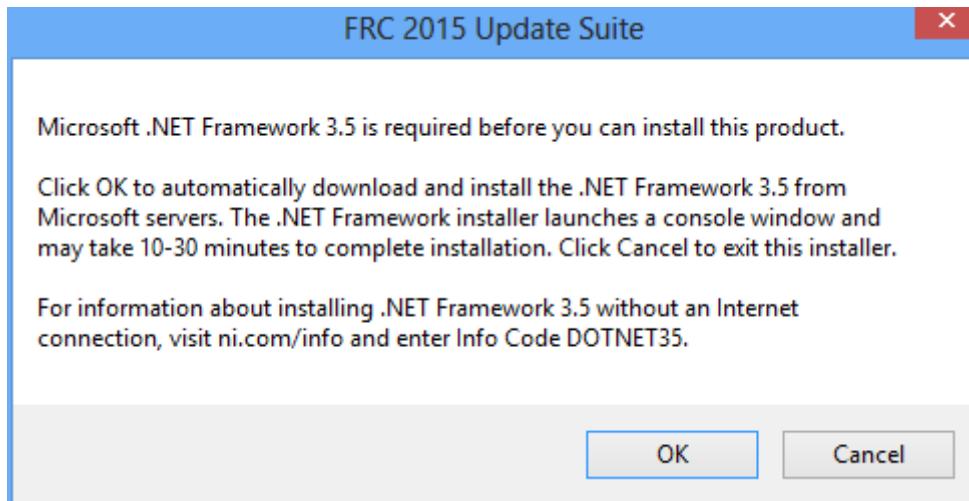
⚠ Note: This download will require the decryption key from the Kickoff broadcast. The key is \$Robots&in#SPACE!!

.NET Framework 4.6.2

The Update installer may prompt that .NET Framework 4.6.2 needs to be updated or installed. Follow prompts on-screen to complete the installation, including rebooting if requested. Then resume the installation of the NI FRC Update, restarting the installer if necessary.

FRC LabVIEW Programming

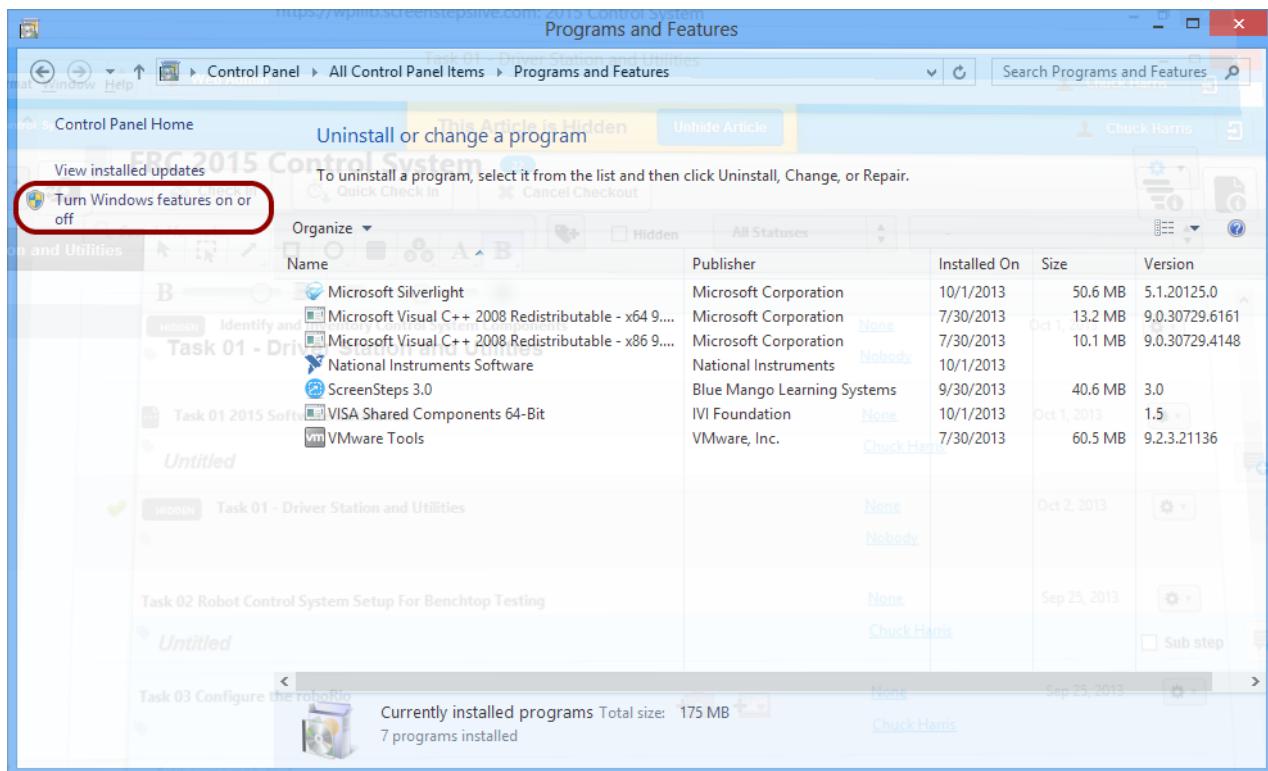
.NET Framework 3.5



If installing on Windows 8 or 10, the Microsoft .NET Framework 3.5 may need to be installed. If you see the dialog shown above, click "Cancel" and perform the steps shown below. An internet connection is required to complete these steps.

FRC LabVIEW Programming

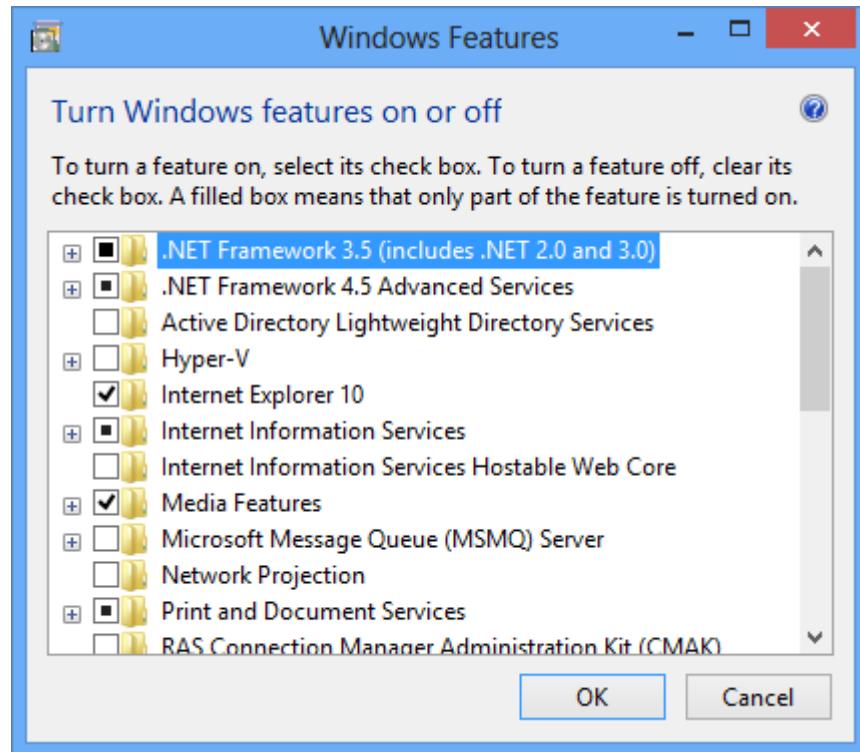
Programs and Features



Open the "Programs and Features" window from the control panel and click on "Turn Windows features on or off"

FRC LabVIEW Programming

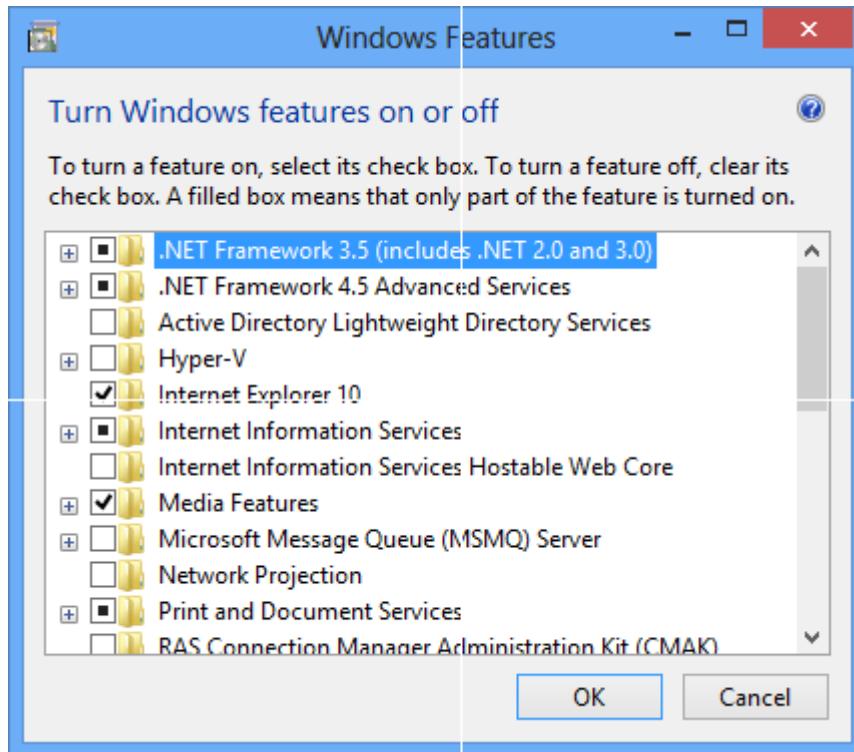
Windows Features (.NET Framework 3.5 not on)



Select ".NET Framework 3.5 (includes .NET 2.0 and 3.0)" to enable it (a black dot, not a check box will appear) and then click "OK". When installation finishes [restart installation of FRC 2019 Update Suite](#).

FRC LabVIEW Programming

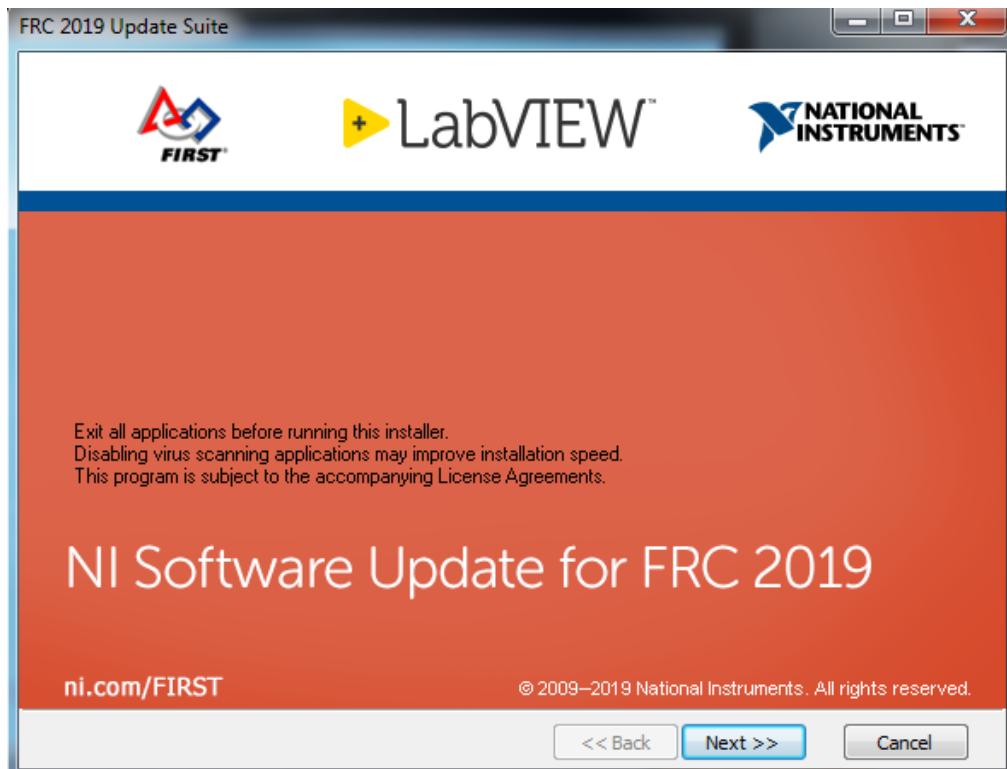
Windows Features (.NET Framework 3.5 already on)



If a black dot is shown next to ".NET Framework 3.5" the feature is already on. Click "Cancel" and [restart installation of FRC 2019 Update Suite](#).

FRC LabVIEW Programming

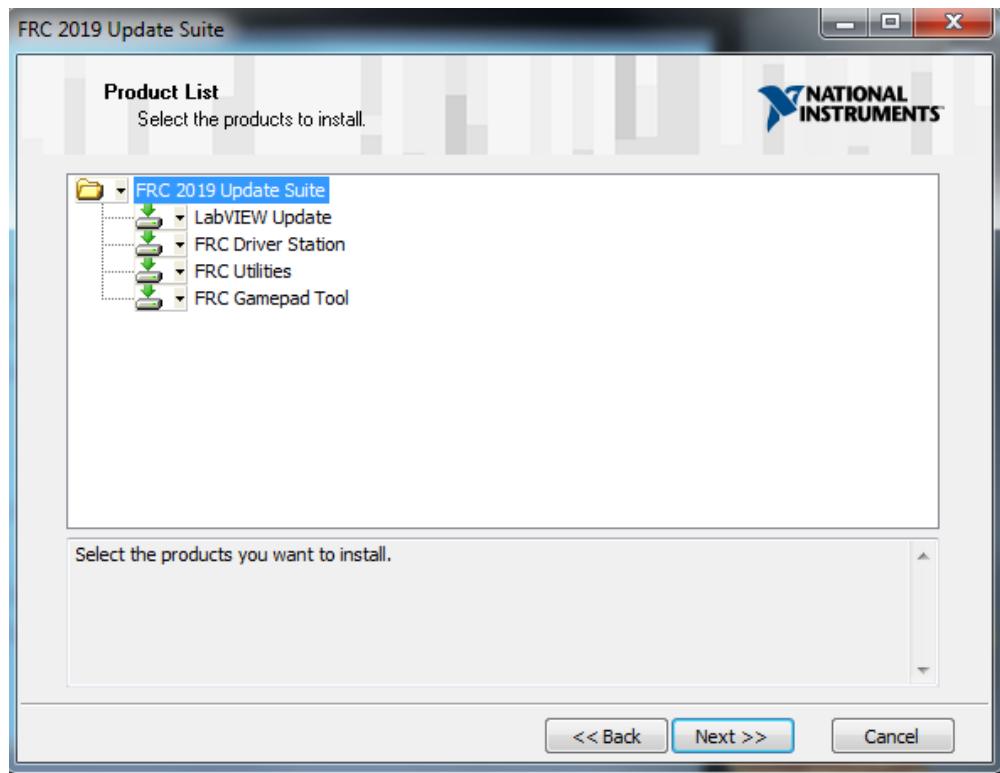
Welcome



Right click on the downloaded zip file and select Extract All. If you downloaded the encrypted zip file, you will be prompted for the encryption key which will be released at Kickoff. Open the extracted folder and any subfolders until you reach the folder containing "setup" (may say "setup.exe" on some machines). Double click on the setup icon to launch the installer. Click "Yes" if a Windows Security prompt appears. Click "Next" on the splash screen that appears.

FRC LabVIEW Programming

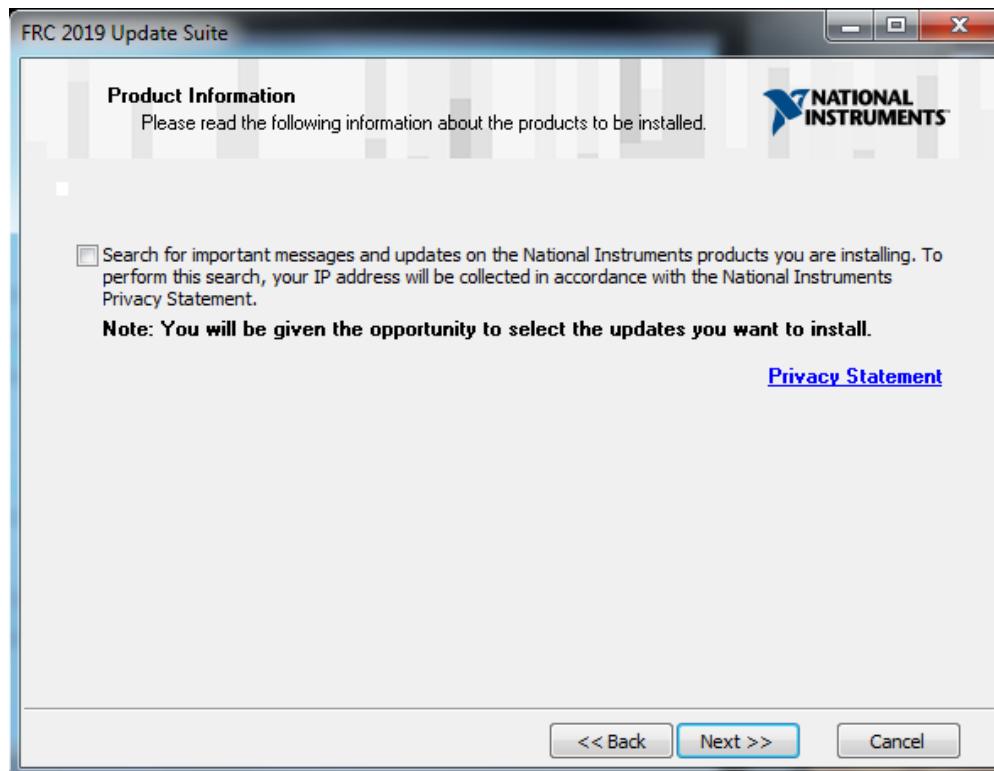
Product List



Click "Next". There is no need to de-select "LabVIEW Update" for C++ or Java teams, if you do not have the base LabVIEW installation (because you are not programming in LabVIEW) this installation will be skipped automatically.

FRC LabVIEW Programming

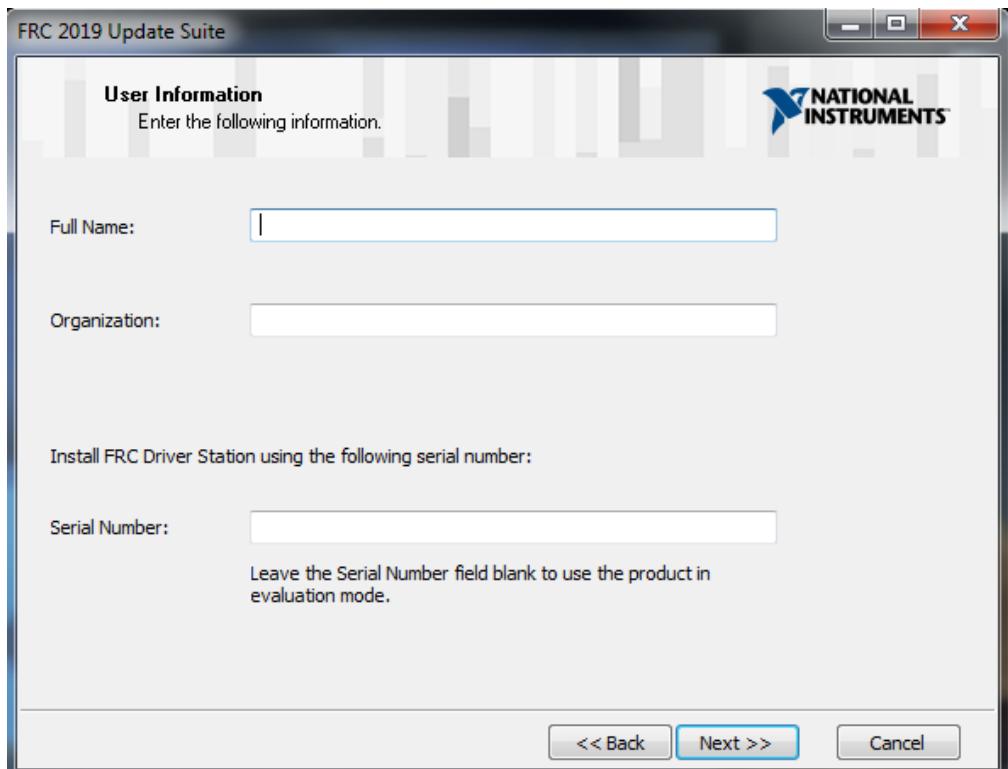
Product Information



Un-check the box, then Click "Next".

FRC LabVIEW Programming

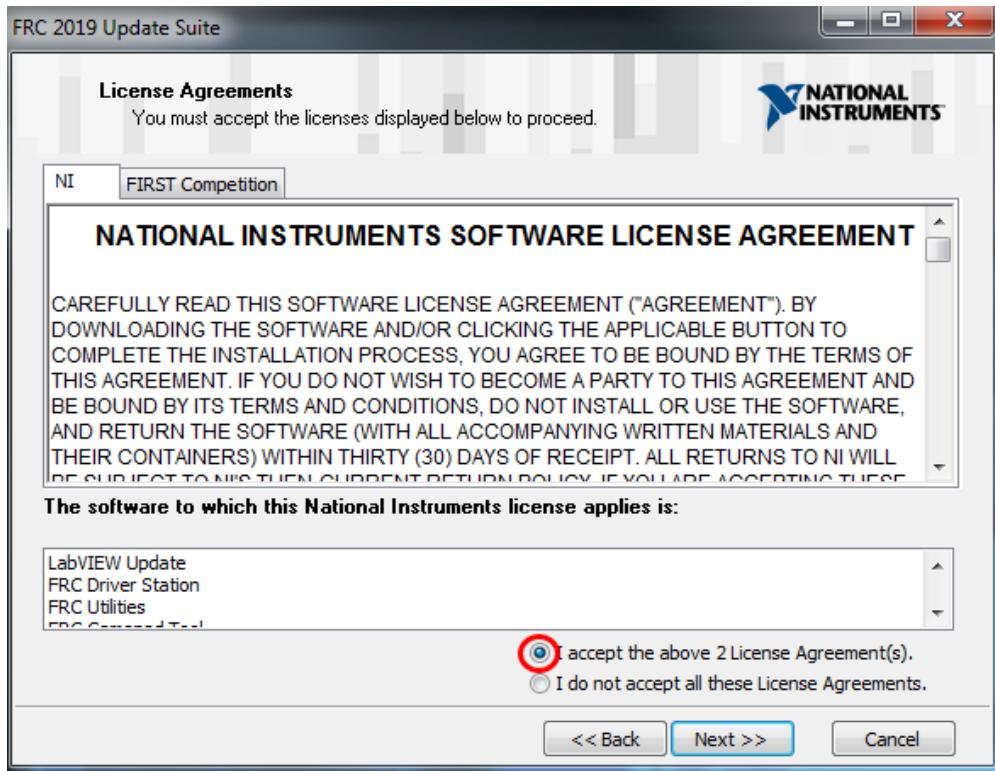
User Information



Enter name and organization and the serial number from the Team Registration System (in the Password/Voucher code section) then click Next

FRC LabVIEW Programming

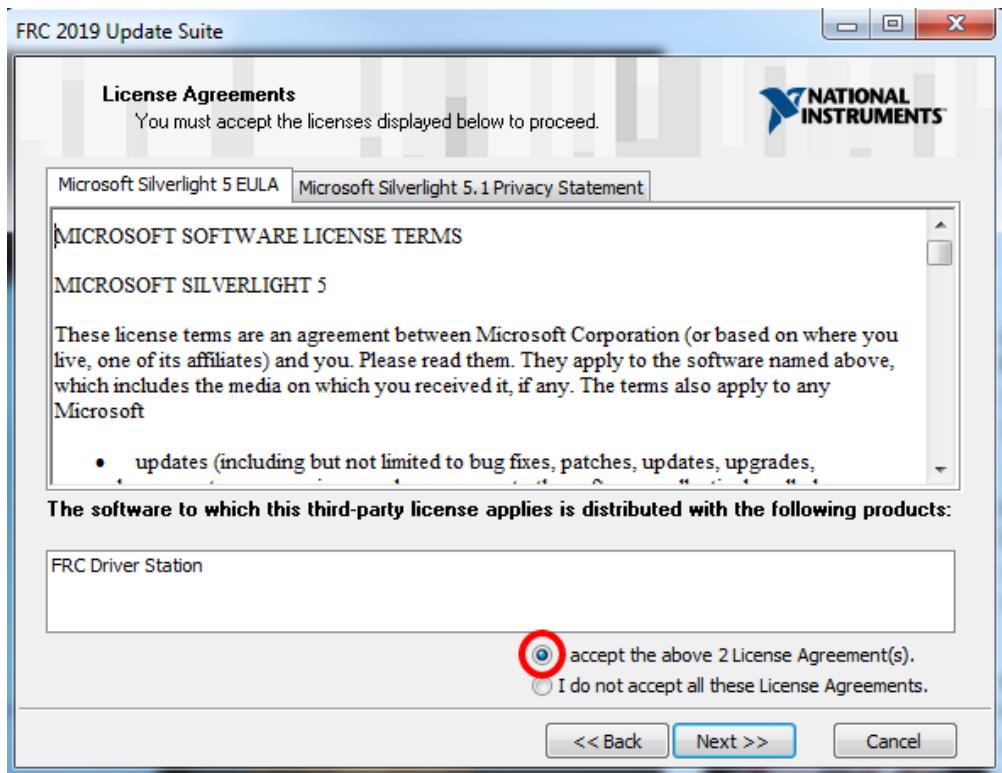
License Agreements



Select "I accept..." then click "Next"

FRC LabVIEW Programming

License Agreements Page 2



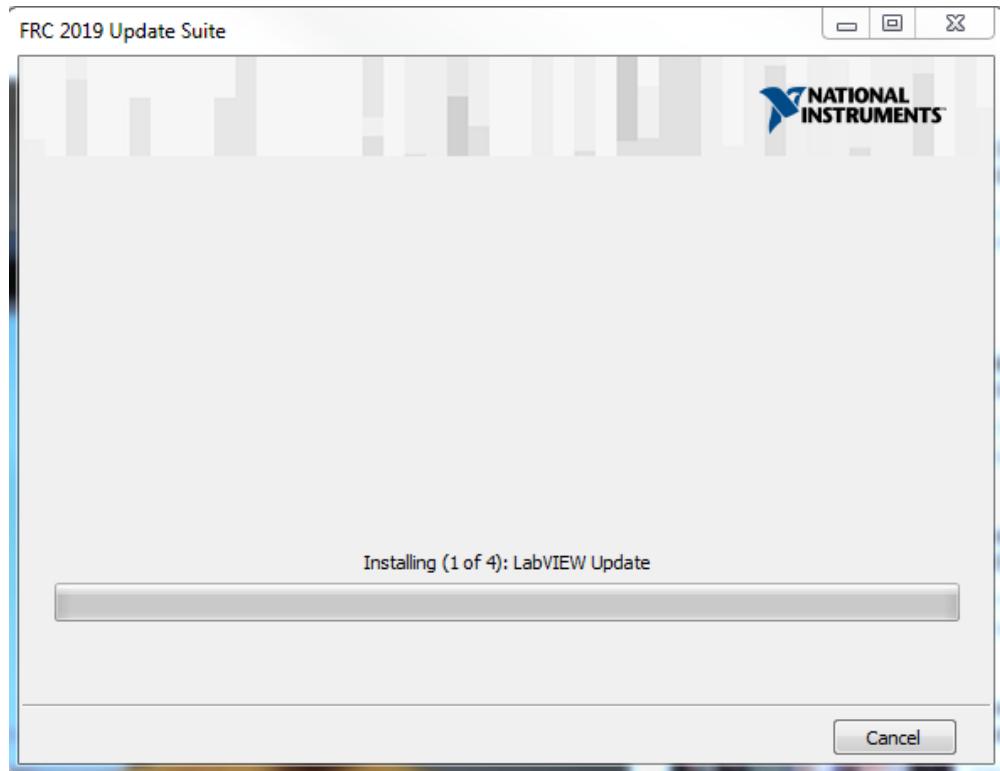
Select "I accept..." then click "Next"

If you see a screen asking to disable Windows Fast Startup, leave it at the recommended option (disable Fast Startup) and click Next.

If you see a screen talking about Windows Firewall, click Next.

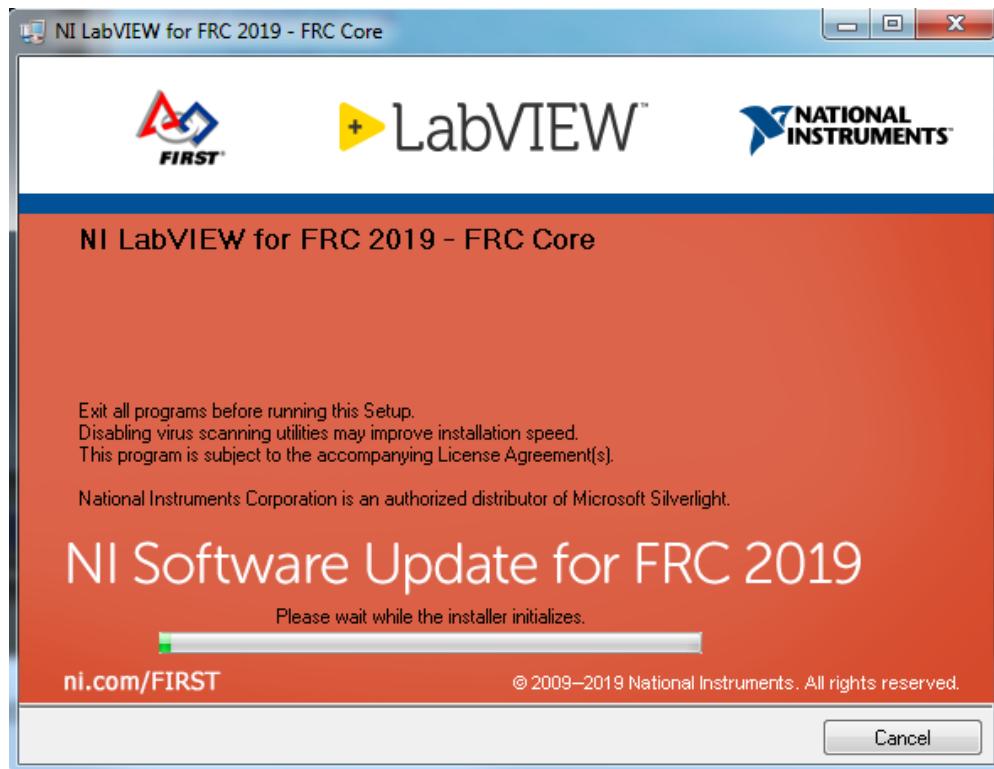
FRC LabVIEW Programming

Summary Progress



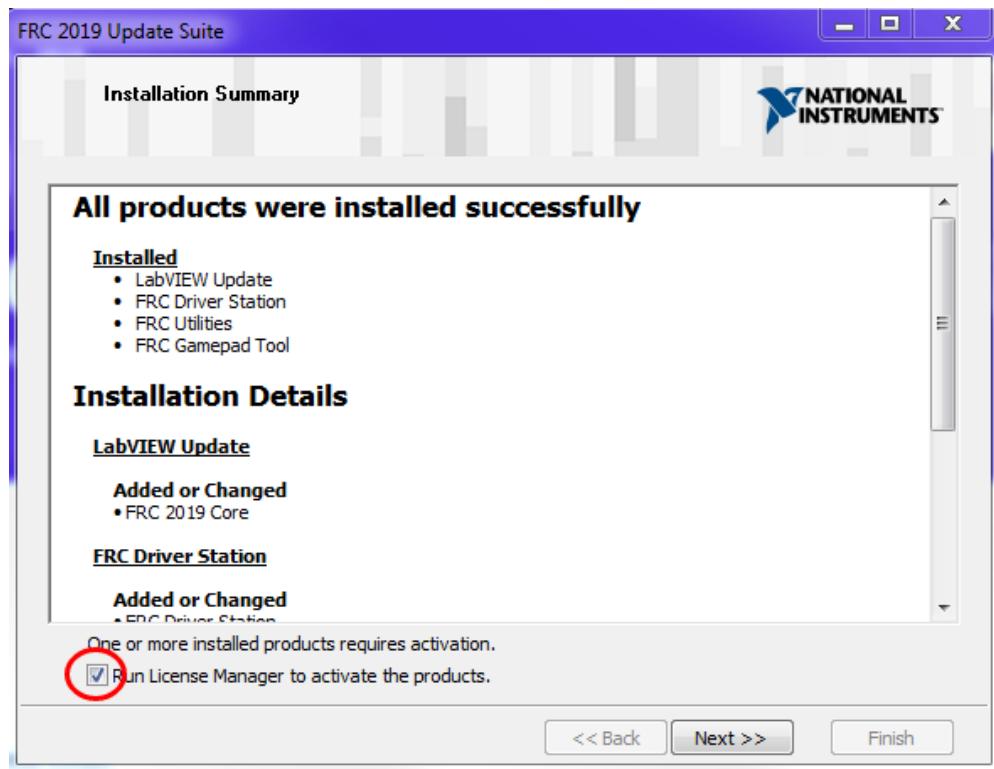
FRC LabVIEW Programming

Detail Progress



FRC LabVIEW Programming

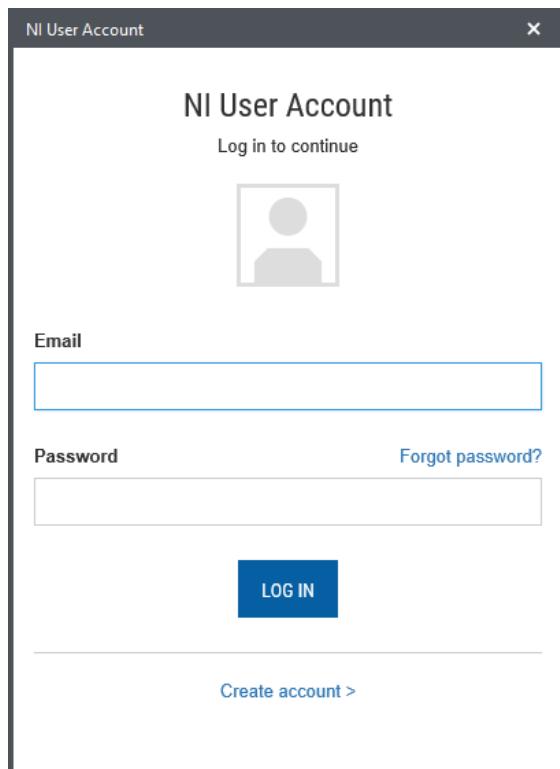
Installation Summary



Make sure the box is checked to Run License Manager... then click Next or Finish

FRC LabVIEW Programming

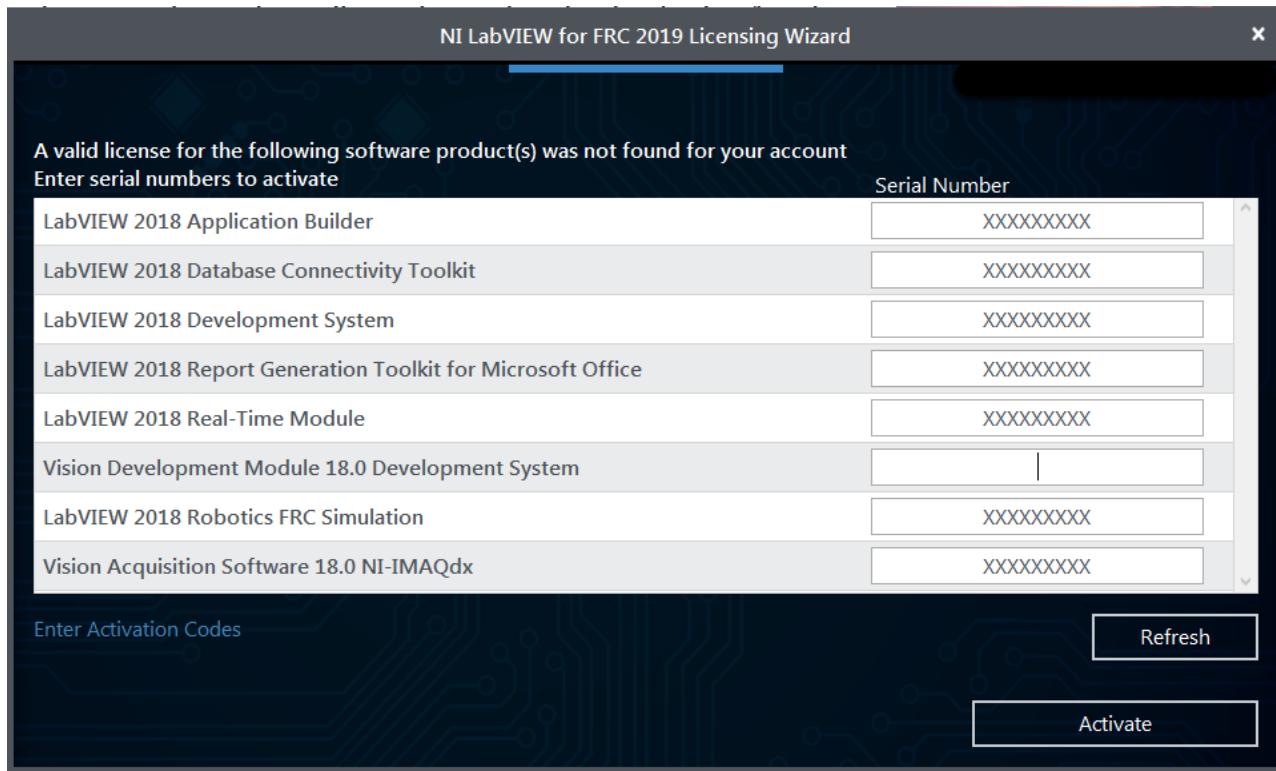
NI Activation Wizard



Log into your ni.com account. If you don't have an account, select 'Create account' to create a free account.

FRC LabVIEW Programming

NI Activation Wizard (2)



The serial number you entered at the "User Information" screen should appear in all of the text boxes, if it doesn't, enter it now. Click "Activate".

Note: If this is the first time activating the 2019 software on this account, you will see the message shown above about a valid license not being found. You can ignore this.

FRC LabVIEW Programming

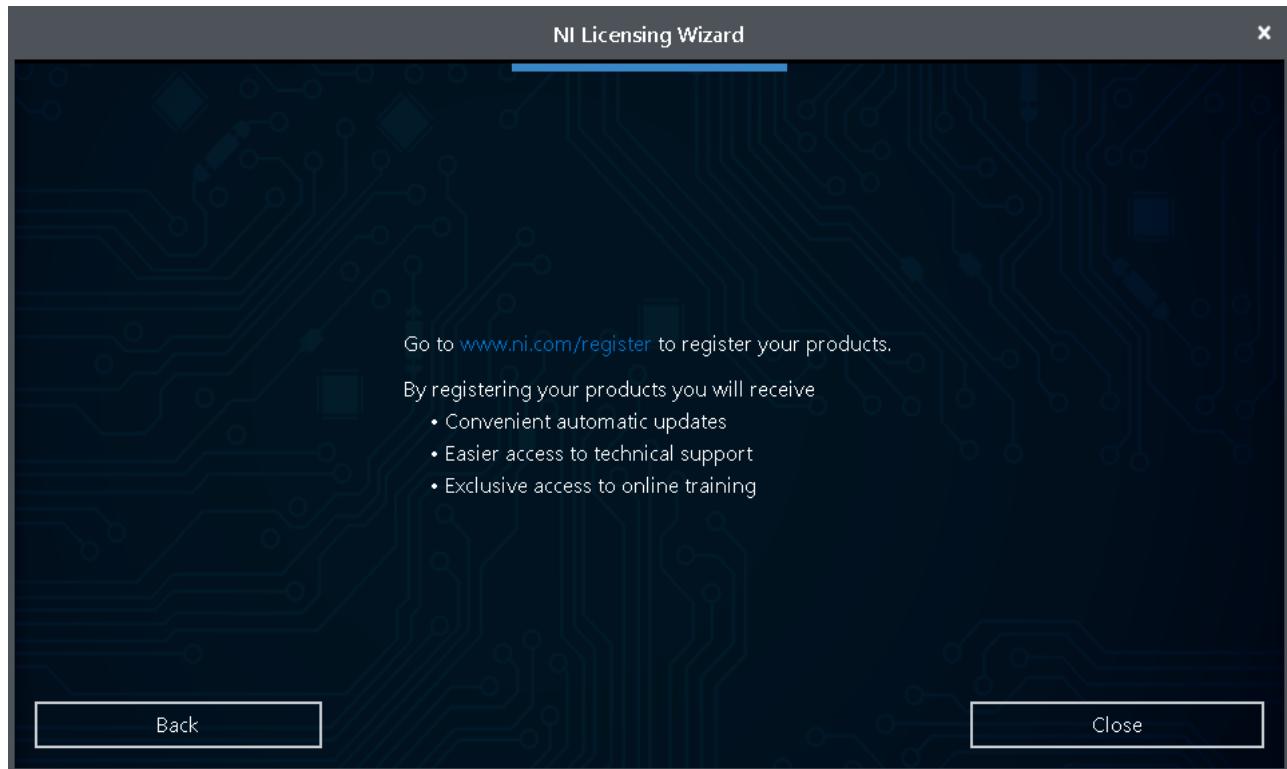
NI Activation Wizard (3)



If your products activate successfully, an “Activation Successful” message will appear. If the serial number was incorrect, it will give you a text box and you can re-enter the number and select “Try Again”. If everything activated successfully, click “Next”.

FRC LabVIEW Programming

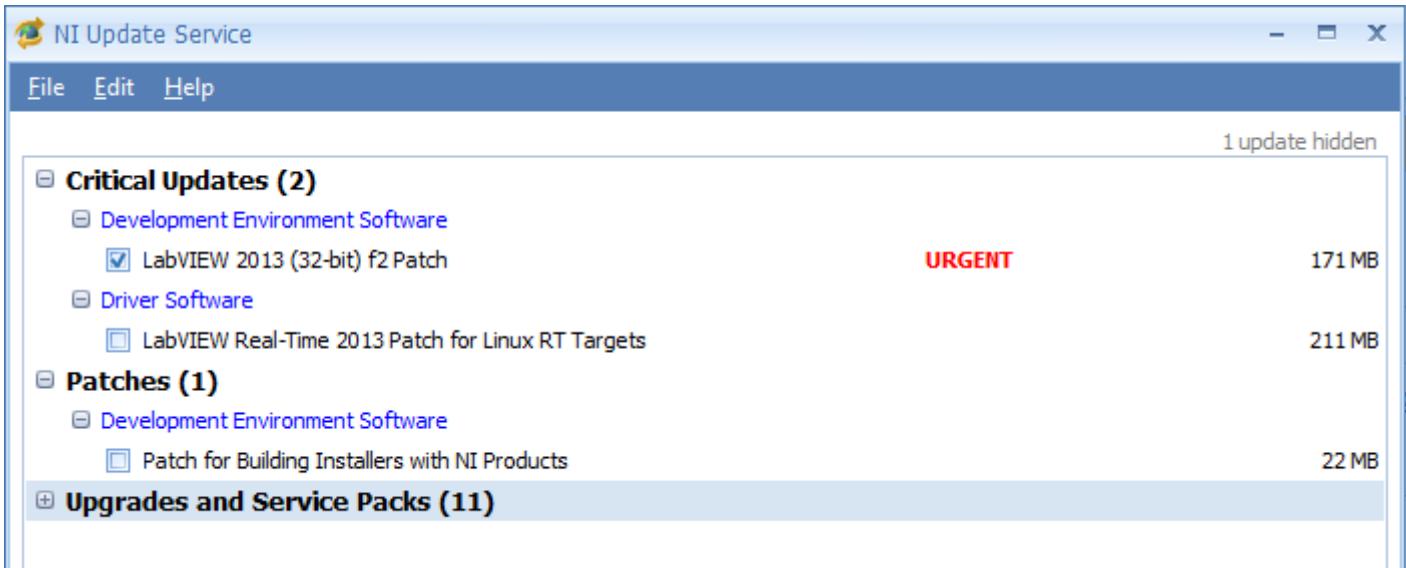
NI Activation Wizard (4)



Click "Close".

FRC LabVIEW Programming

NI Update Service



On occasion you may see alerts from the NI Update Service about patches to LabVIEW. It is not recommended to install these patches. FRC will communicate any recommended updates through our usual channels (Frank's Blog, Team Updates or E-mail Blasts).

Creating Robot Programs

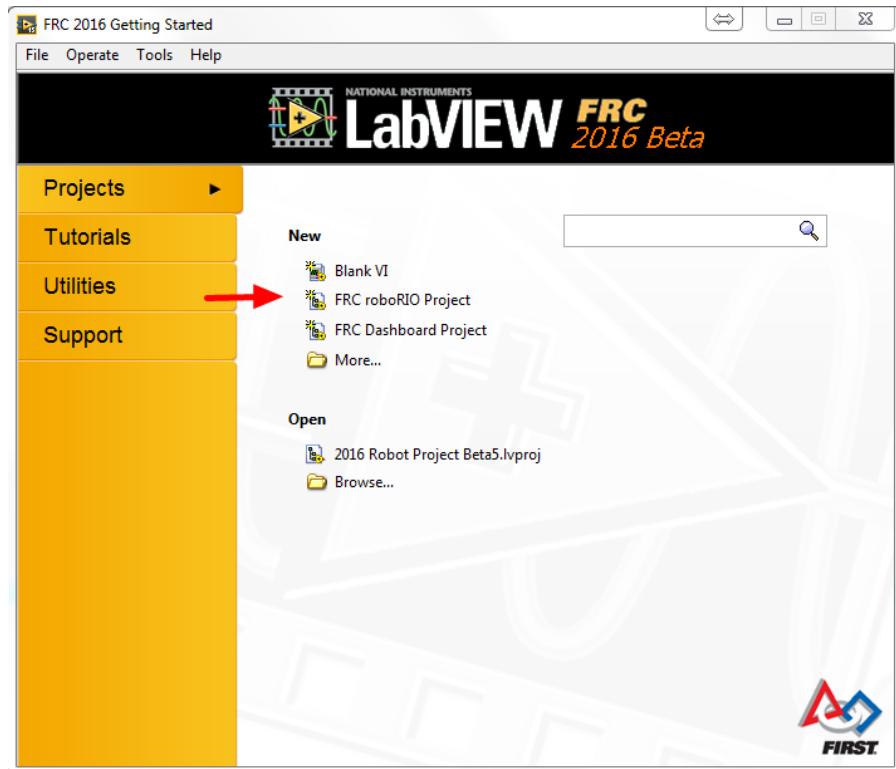
Creating, Building and Loading your Benchtop Test Program



This document covers how to create, build and load an FRC LabVIEW program onto a roboRIO. Before beginning, make sure that you have installed LabVIEW for FRC and the FRC Driver Station and that you have configured and imaged your roboRIO as described in the [Getting Started with the 2016 Control System manual](#).

FRC LabVIEW Programming

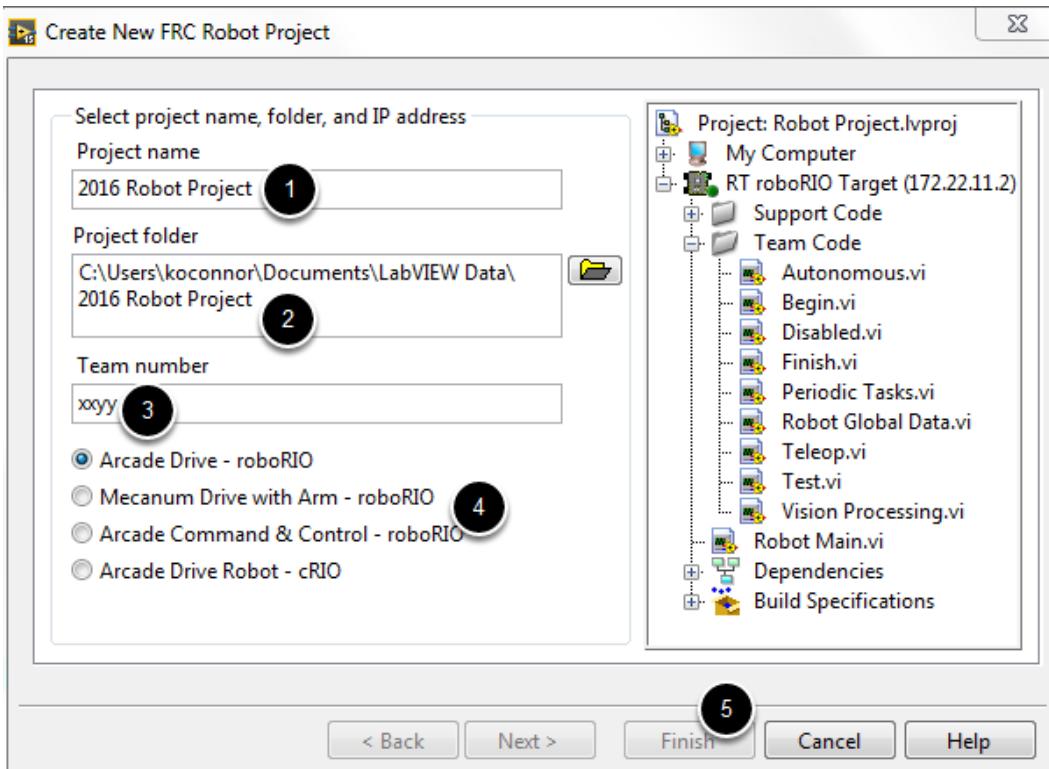
Creating a Project



Launch LabVIEW and click the FRC roboRIO Robot Project link in the Projects window to display the Create New FRC Robot Project dialog box.

FRC LabVIEW Programming

Configuring Project

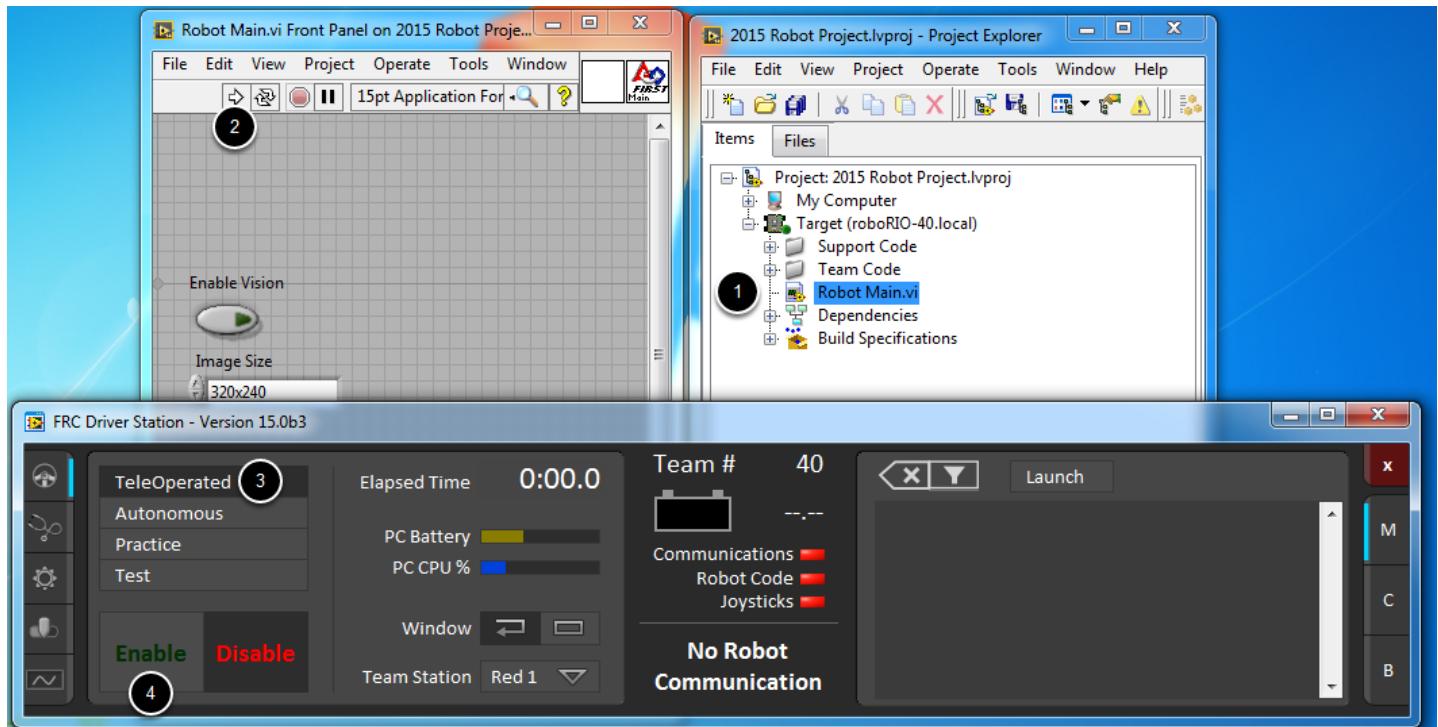


Fill in the Create New FRC Project Dialog:

1. Pick a name for your project
2. Select a folder to place the project in.
3. Enter your team number
4. Select a project type. If unsure, select Arcade Drive - roboRIO.
5. Click Finish

FRC LabVIEW Programming

Running the Program

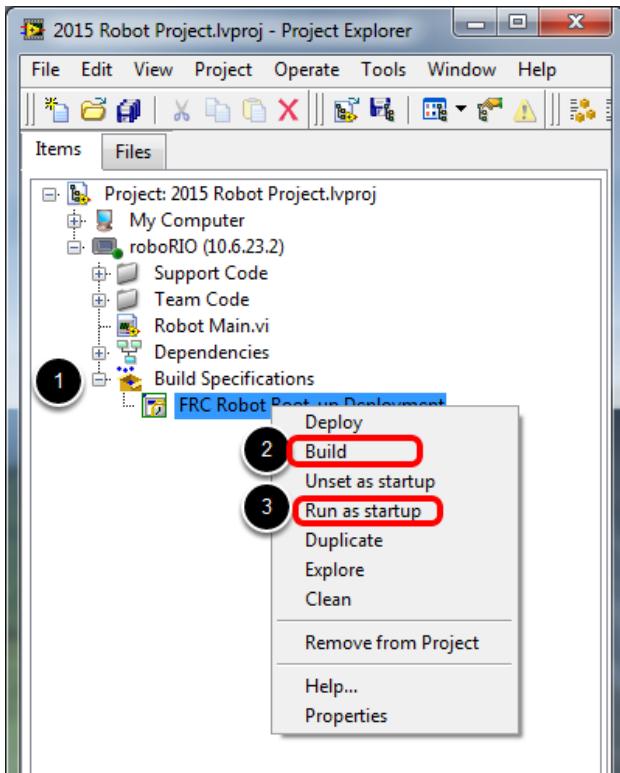


1. In the Project Explorer window, double-click the Robot Main.vi item to open the Robot Main VI.
2. Click the Run button (White Arrow on the top ribbon) of the Robot Main VI to deploy the VI to the roboRIO. LabVIEW deploys the VI, all items required by the VI, and the target settings to memory on the roboRIO. If prompted to save any VIs, click Save on all prompts.
3. Using the Driver Station software, put the robot in Teleop Mode. For more information on configuring and using the Driver Station software, see the [FRC Driver Station Software](#) article.
4. Click Enable.
5. Move the joysticks and observe how the robot responds.
6. Click the Abort button of the Robot Main VI. Notice that the VI stops. When you deploy a program with the Run button, the program runs on the roboRIO, but you can manipulate the front panel objects of the program from the host computer.

Note that a program deployed in this manner will not remain on the roboRIO after a power cycle. To deploy a program to run every time the roboRIO starts follow the next step, Deploying the program.

FRC LabVIEW Programming

Deploying the program



To run in the competition, you will need to deploy a program to your roboRIO. This allows the program to survive across reboots of the controller, but doesn't allow the same debugging features (front panel, probes, highlight execution) as running from the front panel. To deploy your program:

1. In the Project Explorer, click the + next to Build Specifications to expand it.
2. Right-click on FRC Robot Boot-up Deployment and select Build. Wait for the build to complete.
3. Right-click again on FRC Robot Boot-Up Deployment and select Run as Startup. If you receive a conflict dialog, click OK. This dialog simply indicates that there is currently a program on the roboRIO which will be terminated/replaced.
4. Either check the box to close the deployment window on successful completion or click the close button when the deployment completes.
5. The roboRIO will automatically start running the deployed code within a few seconds of the dialog closing.

FRC LabVIEW Programming

Next Steps

To continue Getting Started with the 2016 Control System, see [Programming your radio for home use.](#)

To learn more about LabVIEW, see [LabVIEW Resources](#)

Tank Drive Tutorial

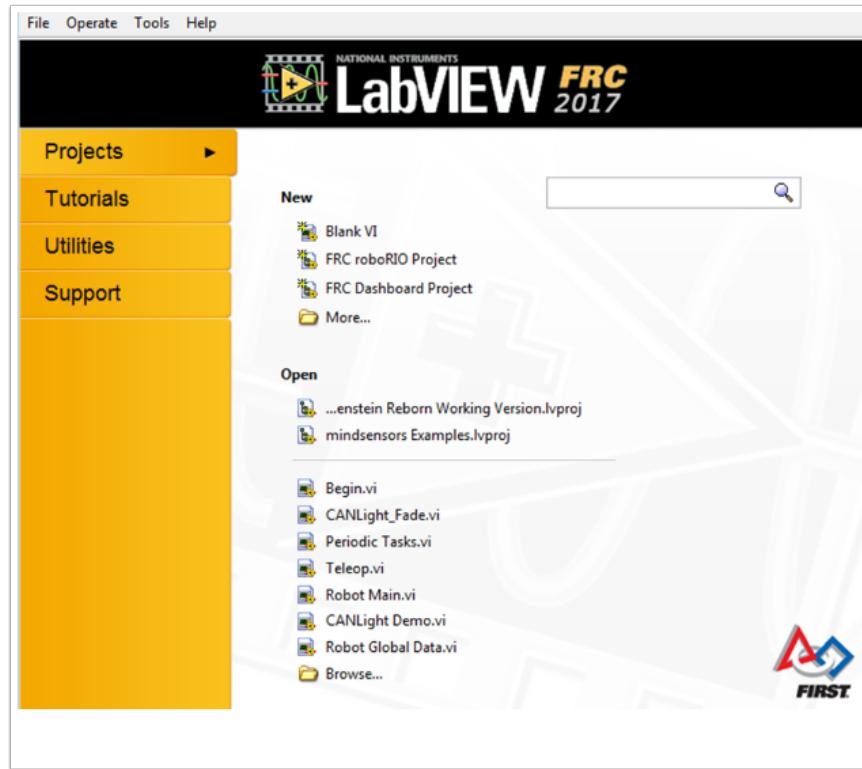


Question: How do I get my robot to drive with two joysticks using tank drive?

Solution: There are four components to consider when setting up tank drive for your robot. The first thing you will want to do is make sure the tank drive.vi is used instead of the arcade drive.vi or whichever drive VI you were utilizing previously. The second item to consider is how you want your joysticks to map to the direction you want to drive. In tank drive, the left joystick is used to control the left motors and the right joystick is used to control the right motors. For example, if you want to make your robot turn right by pushing up on the left joystick and down on the right joystick you will need to set your joystick's accordingly in LabVIEW (this is shown in more detail below). Next, you will want to confirm the PWM lines that you are wired into, are the same ones your joysticks will be controlling. Lastly, make sure your motor controllers match the motor controllers specified in LabVIEW. The steps below will discuss these ideas in more detail:

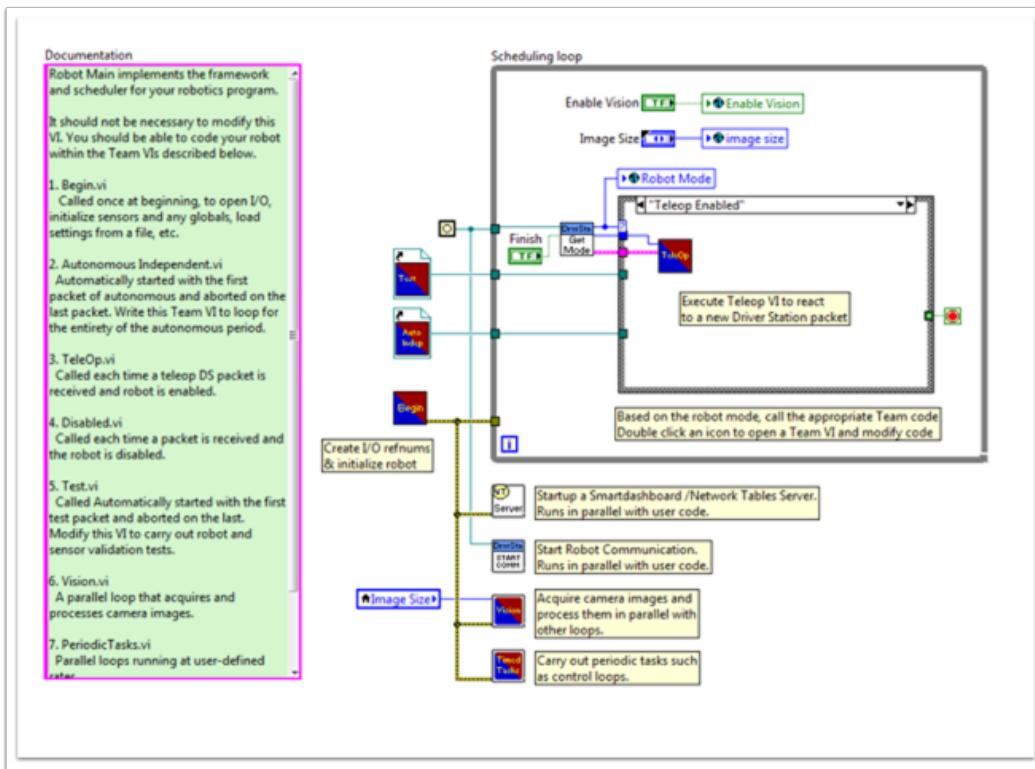
1. Open LabVIEW and double click “FRC roboRIO Project.”

FRC LabVIEW Programming



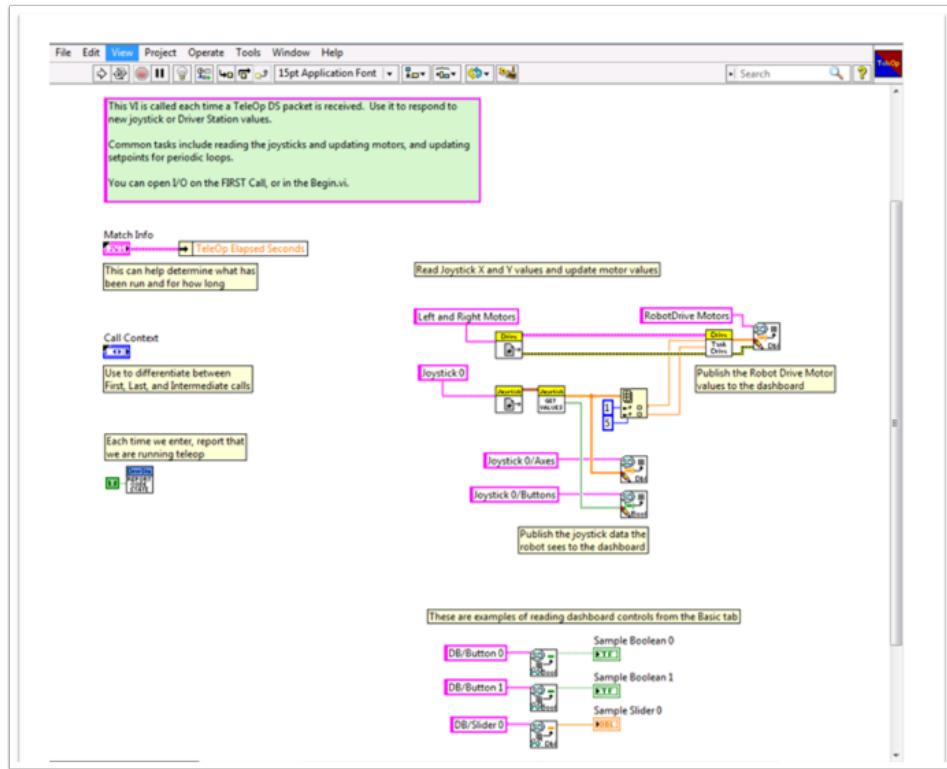
2. Give your project a name, add your team number, and select Arcade Drive Robot roboRIO. You can select another option as well; however, this tutorial will discuss how to setup tank drive for this project.
3. In the Project Explorer window, open up the “Robot Main.vi.”
4. Push Ctrl + E to see the block diagram. It should look like the following image:

FRC LabVIEW Programming



- Double click the “Teleop” vi inside of the Teleop Enabled case structure. Look at its block diagram. You will want to make two changes here:
 - Replace Arcade Drive with the tank drive.vi. This can be found by right clicking on the block diagram >> WPI Robotics Library >> Robot Drive >> and clicking the Tank Drive VI.
 - Find the Index Array function that is after the Get Values.vi. You will need to create two numeric constants and wire each into one of the index inputs. You can determine what the values of each index should be by looking at the USB Devices tab in the FRC Driver Station. Move the two joysticks to determine which number (index) they are tied to. You will likely want to use the Y-axis index for each joystick. This is because it is intuitive to push up on the joystick when you want the motors to go forward, and down when you want them to go in reverse. If you select the X-axis index for each, then you will have to move the joystick left or right (x-axis directions) to get the robot motors to move. In my setup, I've selected index 1 for my left motors Y-axis control and index 5 as the right motors Y-axis control. You can see the adjustments in LabVIEW in the following image:

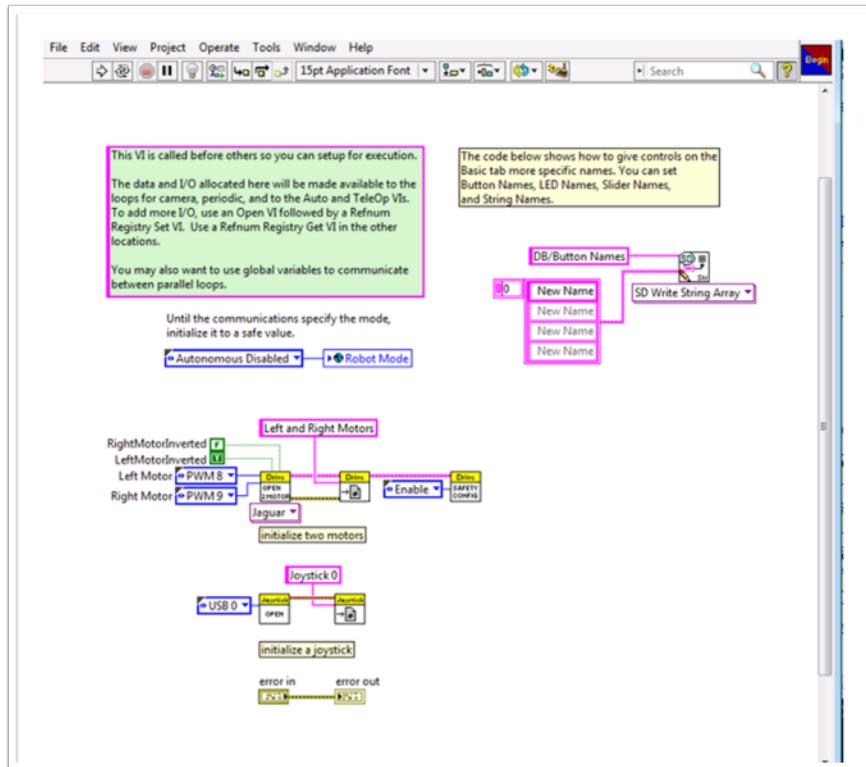
FRC LabVIEW Programming



6. Next you will want to go back to your “Robot Main.vi” and double click on the “Begin.vi.”
7. The first thing to confirm in this VI is that your left and right motors are connected to the same PWM lines in LabVIEW as they are on your PDP (Power Distribution Panel).
8. The second thing to confirm in this VI is that the “Open 2 Motor.vi” has the correct motor controller selected (Talon, Jaguar, Victor, etc.).

For example, I am using Jaguar motor controllers and my motors are wired into PWM 8 and 9. The image below shows the changes I need to make:

FRC LabVIEW Programming



9. Save all of the Vis that you have made adjustments to and you are now able to drive a robot with tank drive!

Command and Control Tutorial



Introduction

Command and Control is a new LabVIEW template added for the 2016 season which organizes robot code into commands and controllers for a collection of robot-specific subsystems. Each subsystem has an independent control loop or state machine running at the appropriate rate for the mechanism and high-level commands that update desired operations and set points. This makes it very easy for autonomous code to build synchronous sequences of commands. Meanwhile, TeleOp benefits because it can use the same commands without needing to wait for completion, allowing for easy cancellation and initiation of new commands according to the drive team input. Each subsystem has a panel displaying its sensor and control values over time, and command tracing to aid in debugging.

What is Command and Control?

Command and Control recognizes that FRC robots tend to be built up of relatively independent mechanisms such as Drive, Shooter, Arm, etc. Each of these is referred to as a subsystem and needs code that will coordinate the various sensors and actuators of the subsystem in order to complete requested commands, or actions, such as "Close Gripper" or "Lower Arm". One of the key principles of this framework is that subsystems will each have an independent controller loop that is solely responsible for updating motors and other actuators. Code outside of the subsystem controller can issue commands which may change the robot's output, but should not directly change any outputs. The difference is very subtle but this means that outputs can only possibly be updated from one location in the project. This speeds up debugging a robot behaving unexpectedly by giving you the ability to look through a list of commands sent to the subsystem rather than searching your project for where an output may have been modified. It also becomes easier to add an additional sensor, change gearing, or disable a mechanism without needing to modify code outside of the controller.

Game code, primarily consisting of Autonomous and TeleOp, will typically need to update set points and react to the state of certain mechanisms. For Autonomous, it is very common to define the robot's operation as a sequence of operations – drive here, pick that up, carry it there, shoot it, etc. Commands can be wired sequentially with additional logic to quickly build complex routines. For teleOp, the same commands can execute asynchronously, allowing the robot to always process the latest driver inputs, and if implemented properly, new commands will interrupt,

FRC LabVIEW Programming

allowing the drive team to quickly respond to field conditions while also taking advantage of automated commands and command sequences.

Why should I use Command and Control?

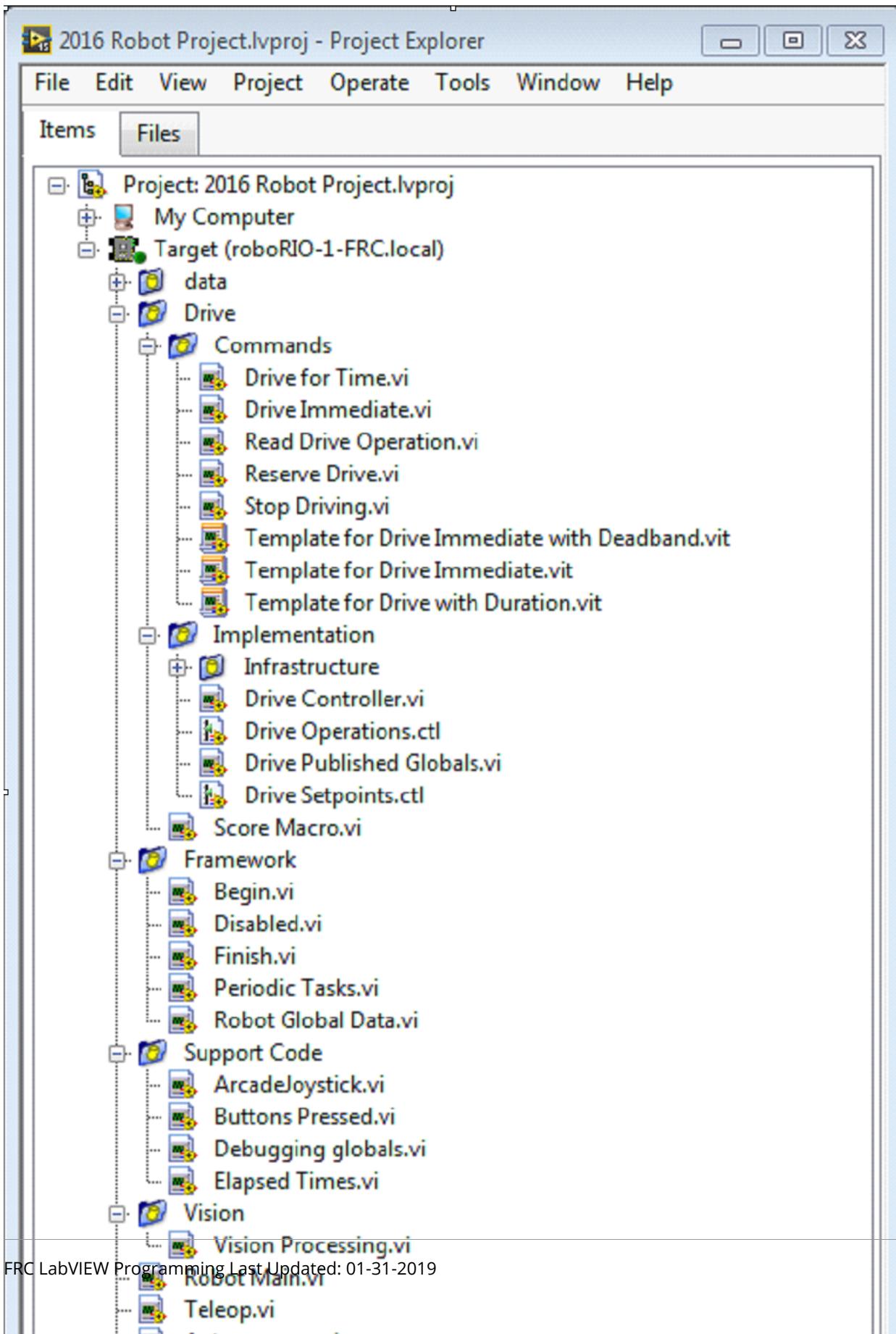
Command and Control adds functionality to the existing LabVIEW project templates, allowing code to scale better with more sophisticated robots and robot code. Subsystems are used to abstract the details of the implementation, and game code is built from sequences of high level command VIs. The commands themselves are VIs that can update set points, perform numerical scaling/mapping between engineering units and mechanism units, and offer synchronization options. If physical changes are made to the robot, such as changing a gearing ratio, changes can be made to just a few command Vis to reflect this change across the entire code base.

I/O encapsulation makes for more predictable operation and quicker debugging when resource conflicts do occur. Because each command is a VI, you are able to single step through commands or use the built in Trace functionality to view a list of all commands sent to each subsystem. The framework uses asynchronous notification and consistent data propagation making it easy to program a sequence of commands or add in simple logic to determine the correct command to run.

Part 1: Project Explorer

The Project Explorer provides organization for all of the Vis and files you will use for your robot system. Below is a description of the major components in the Project Explorer to help with the expansion of our system. The most frequently used items have been bolded.

FRC LabVIEW Programming



FRC LabVIEW Programming

My Computer – the items that define operation on the computer that the project was loaded on. For a robot project, this is used as a simulation target and is populated with simulation files.

Sim Support Files — folder containing 3D CAD models and description files for the simulated robot.

Robot Simulation Readme.html — documents the PWM channels and robot info you will need in order to write robot code that matches the wiring of the simulated robot.

Dependencies – shows the files used by the simulated robot's code. This will populate when you designate the code for the simulated robot target.

Build Specifications – will contain the files that define how to build and deploy code for the simulated robot target.

Target (roboRIO-TEAM-FRC.local) – the items that define operation on the roboRIO located at (address).

Drive — subsystem implementation and commands for the robot drive base. This serves as a custom replacement for the WPILib RobotDrive VIs.

Framework — VIs used for robot code that is not part of a subsystem that are not used very often.

- Begin — called once when robot code first starts. This is useful for initialization code that doesn't belong to a particular subsystem.
- Disabled — called once for each disabled packet and can be used to debug sensors when you don't want the robot to move.
- Finish — during development, this may be called when robot code finishes. Not called on abort or when power is turned off.
- Periodic Tasks — a good place for ad hoc periodic loops for debugging or monitoring
- Robot Global Data — useful for sharing robot information that doesn't belong to a subsystem.

FRC LabVIEW Programming

Support Code — debugging and code development aids.

Vision — subsystem and commands for the camera and image processing.

Robot Main.vi — Top level VI that you will run while developing code.

Autonomous.vi — VI that runs during autonomous period.

Teleop.vi — VI that is called for each TeleOp packet.

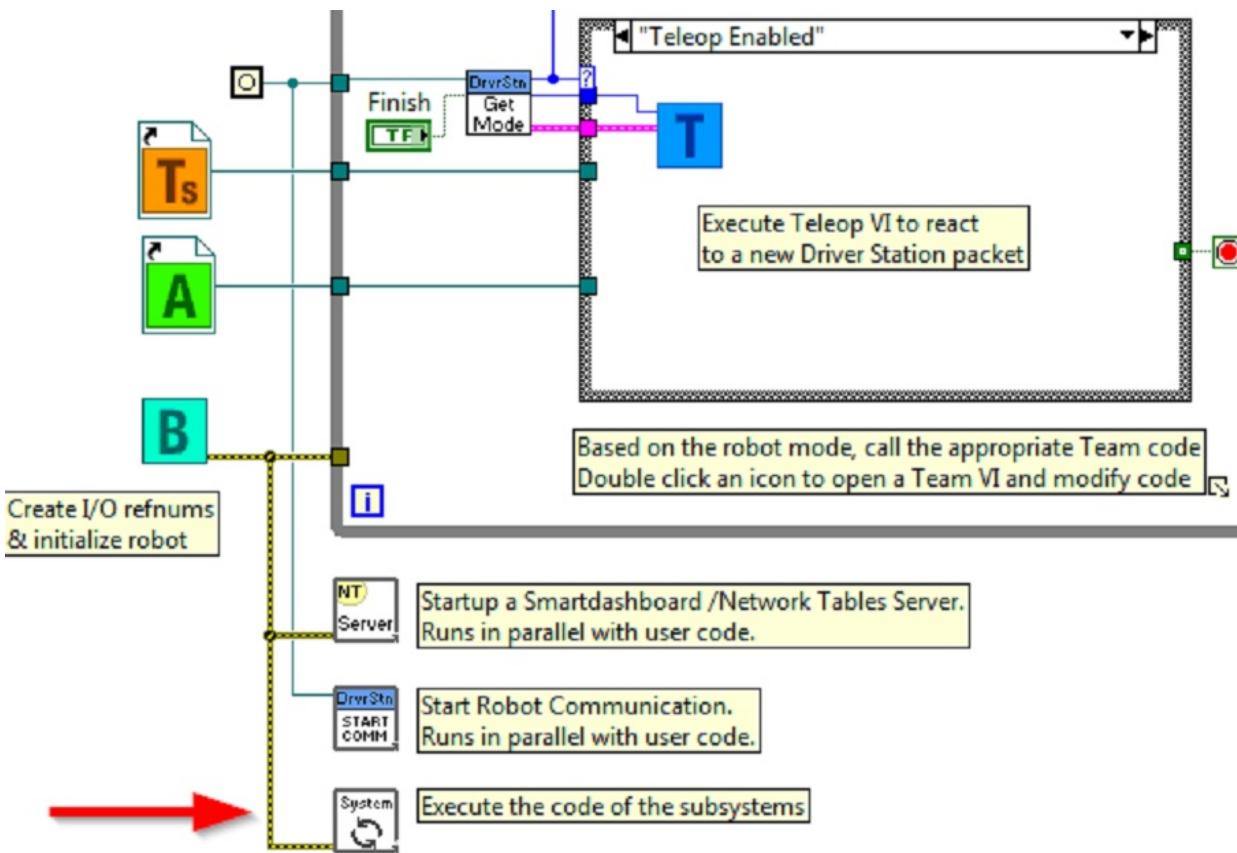
Test.vi — VI that runs when driver station is in test mode.

SubSystems.vi — VI that contains and starts all subsystems.

Dependencies – shows the files used by the robot code.

Build Specifications — used to build and run the code as a startup application once code works correctly.

FRC LabVIEW Programming



Drive Subsystem Project Explorer

Commands:

This folder contains the command VIs that request the controller carry out an operation. It also contains templates for creating additional drive commands.

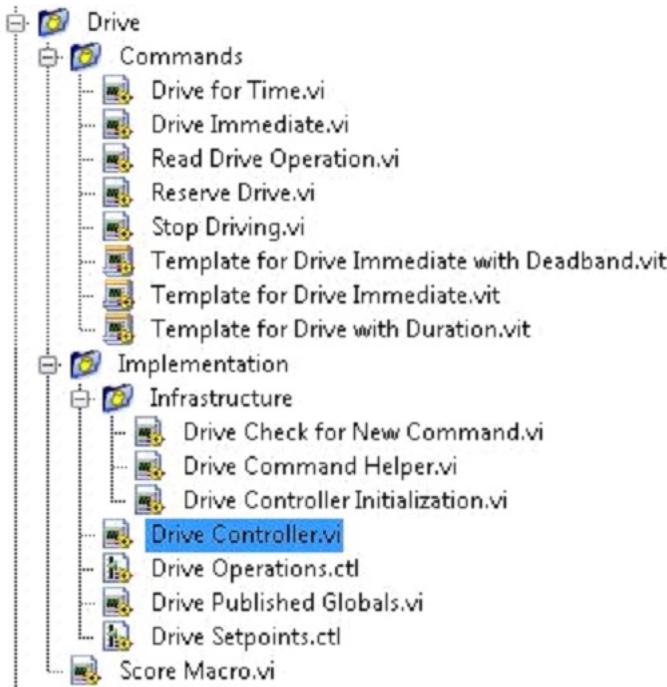
Note: After creating a new command, you may need to edit Drive Setpoints.ctl to add or update fields that controller uses to define the new operation. You also need to go into the Drive Controller.vi and modify the case structure to add a case for every value.

Implementation:

- These are the VIs and Controls used to build the subsystem.
- Infrastructure VIs
 - Drive Check for New Command: It is called each iteration of the controller loop. It checks for new commands, updates timing data, and upon completion notifies a waiting command.
 - Drive Command Helper.vi: Commands call this VI to notify the controller that a new command has been issued.

FRC LabVIEW Programming

- Drive Controller Initialization.vi: It allocates the notifier and combines the timing, default command, and other information into a single data wire.
- Drive Controller.vi: This VI contains the control/state machine loop. The panel may also contain displays useful for debugging.
- Drive Operation.ctl: This typedef defines the operational modes of the controller. Many commands can share an operation.
- Drive Setpoint.ctl: It contains the data fields used by all operating modes of the Drive subsystem.
- Drive Published Globals.vi: A useful place for publishing global information about the drive subsystem.

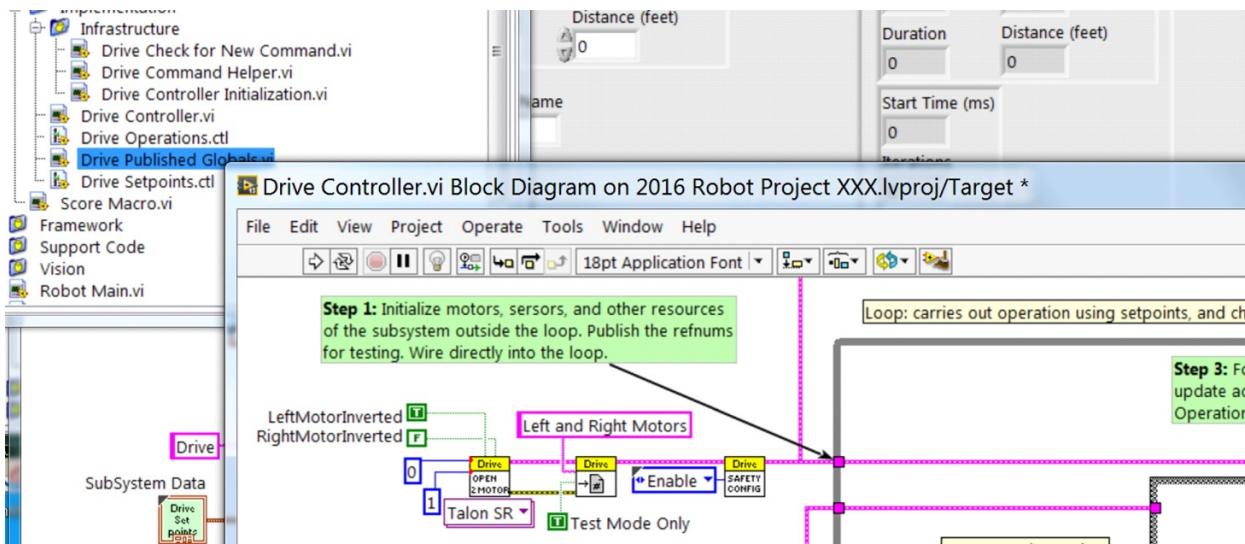


Part 2: Initializing the Drive Subsystem

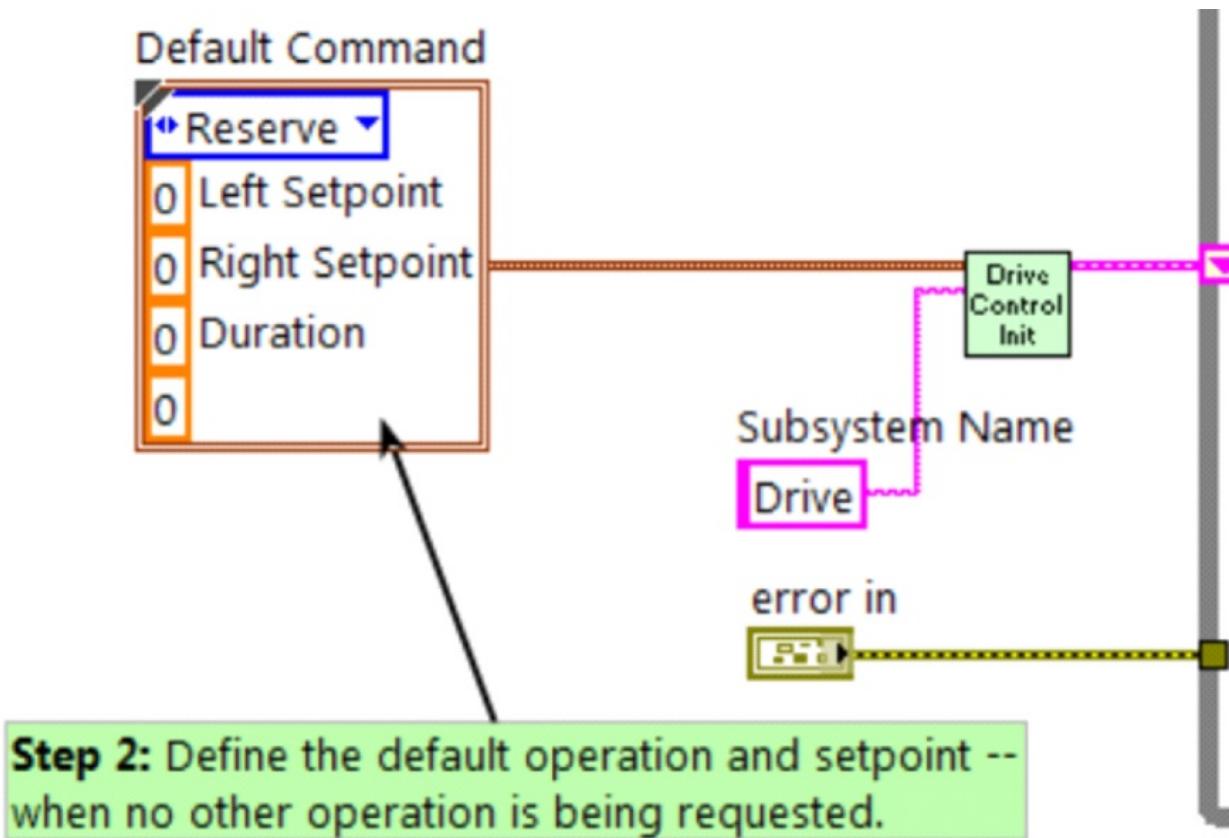
There are green comments on the controller's block diagram that point out key areas that you will want to know how to edit.

The area to the left of the control loop will execute once when the subsystem starts up. This is where you will typically allocate and initialize all I/O and state data. You may publish the I/O refnums, or you may register them for Test Mode Only to keep them private so that other code cannot update motors without using a command.

FRC LabVIEW Programming

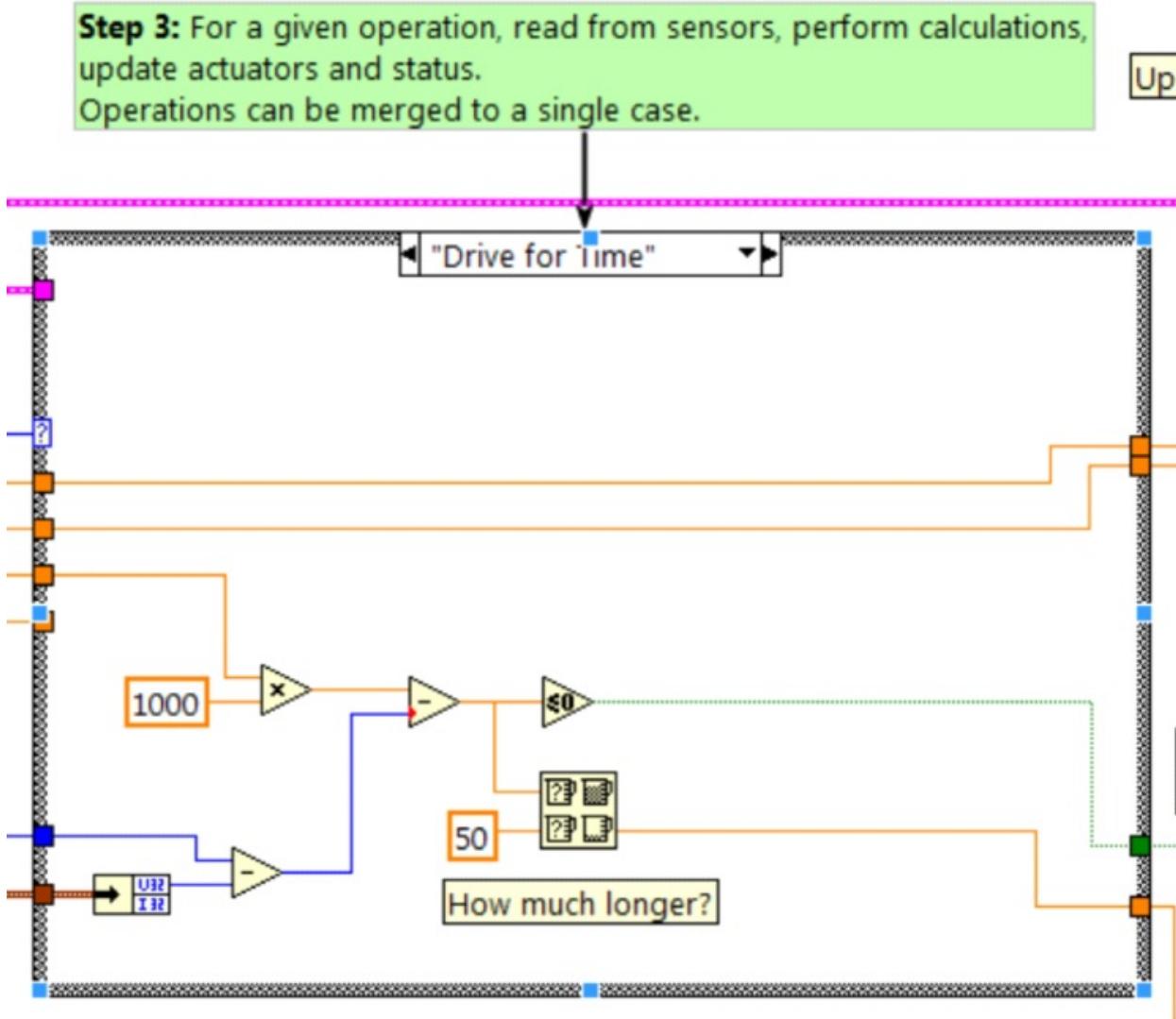


Note: Initializing the resources for each subsystem in their respective Controller.vi rather than in Begin.vi improves I/O encapsulation, reducing potential resource conflicts and simplifies debugging.



FRC LabVIEW Programming

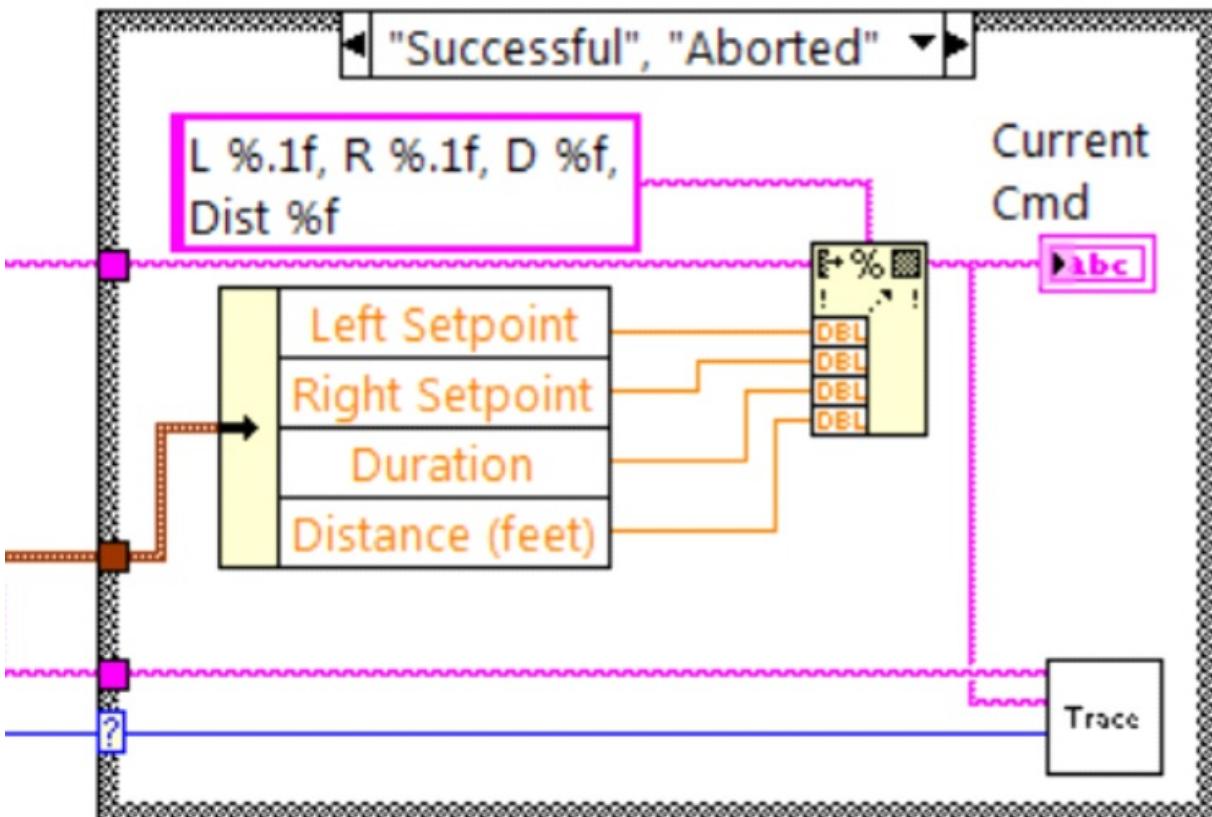
Part of the initialization is to select the default operation and set point values when no other operation is being processed.



Inside the control loop is a case statement where operations are actually implemented. Set point values, iteration delay, iteration count, and sensors can all have influence on how the subsystem operates. This case structure has a value for each operation state of the subsystem.

FRC LabVIEW Programming

Step 4: Format a string to describe this cmd and how the previous cmd finished



Each iteration of the controller loop will optionally update the Trace VI. The framework already incorporates the subsystem name, operation, and description, and you may find it helpful to format additional set point values into the trace information. Open the Trace VI and click Enable while the robot code is running to current setpoints and commands sent to each subsystem.

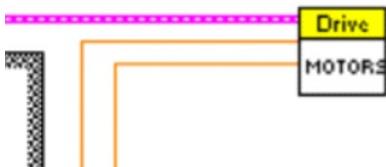
The primary goal of the controller is to update actuators for the subsystem. This can occur within the case structure, but many times, it is beneficial to do it downstream of the structure to ensure that values are always updated with the correct value and in only one location in the code.

FRC LabVIEW Programming

Step 5: Update your I/O

Update motors and update dashboard

RobotDrive Motors



Part 3: Drive Subsystem Shipped Commands

There are 3 shipped example commands for each new subsystem:

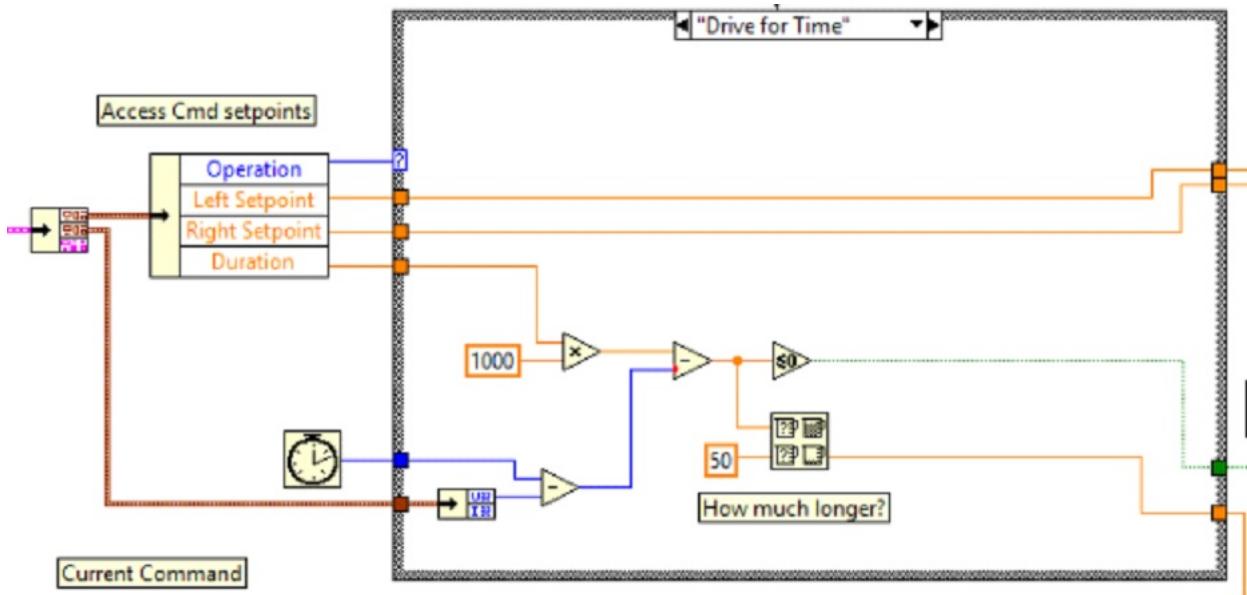
Drive For Time.vi



This VI sets the motors to run for a given number of seconds. It optionally synchronizes with the completion of the command.

The Drive for Time case will operate the motors at the set point until the timer elapses or a new command is issued. If the motors have the safety timeout enabled, it is necessary to update the motors at least once every 100ms. This is why the code waits for the smaller of the remaining time and 50ms.

FRC LabVIEW Programming



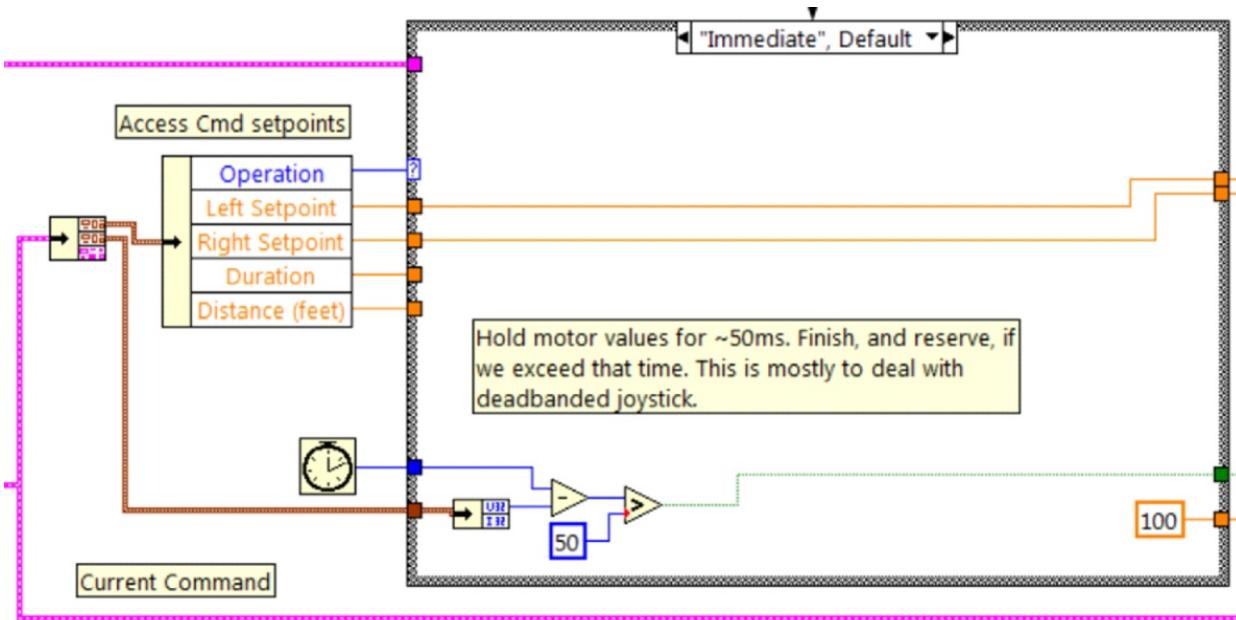
Drive Immediate.vi



Gets the desired left and right speeds for the motors and will set the motors immediately to those set points.

The Immediate case updates the motors to the set point defined by the command. The command is not considered finished since you want the motors to maintain this value until a new command comes in or until a timeout value. The timeout is useful anytime a command includes a dead band. Small values will not be requested if smaller than the dead band, and will result in growling or creeping unless the command times out.

FRC LabVIEW Programming



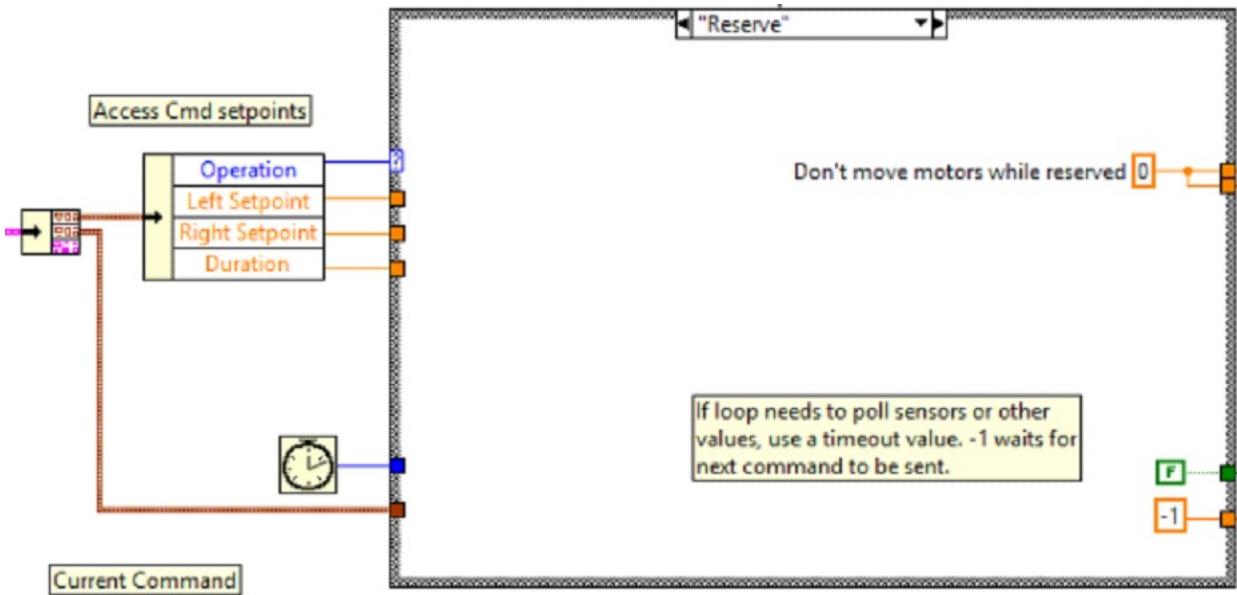
Stop Driving.vi



Zero the drive motors, making the robot stationary.

The Reserve command turns off the motors and waits for a new command. When used with a named command sequence, reserve identifies that the drive subsystem is part of a sequence, even if not currently moving the robot. This helps to arbitrate subsystem resource between simultaneously running commands.

FRC LabVIEW Programming



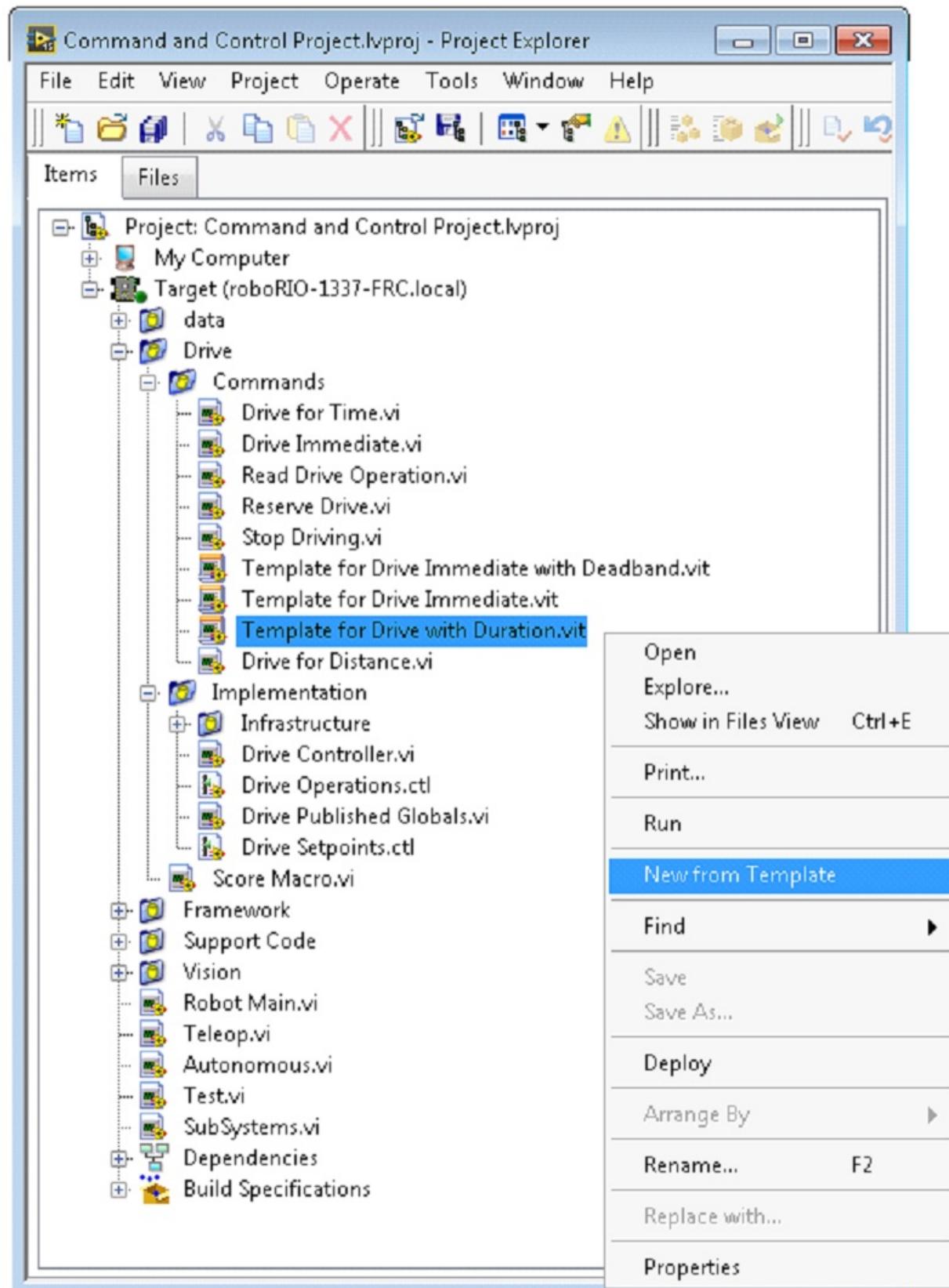
Part 4: Creating New Commands

The Command and Control framework allows users to easily create new commands for a subsystem. To Create a new command open the subsystem folder/Commands In the project explorer window, choose one of the VI Templates to use as the starting point of your new command, right click, and select New From Template.

- Immediate: This VI notifies the subsystem about the new setpoint.
- Immediate with deadband: This VI compares the input value to the deadband and optionally notifies the subsystem about the new setpoint. This is very useful when joystick continuous values are being used.
- With duration: This VI notifies the subsystem to perform this command for the given duration, and then return to the default state. Synchronization determines whether this VI Starts the operation and returns immediately, or waits for the operation to complete. The first option is commonly used for TeleOp, and the second for Autonomous sequencing.

In this example we will add the new command "Drive for Distance".

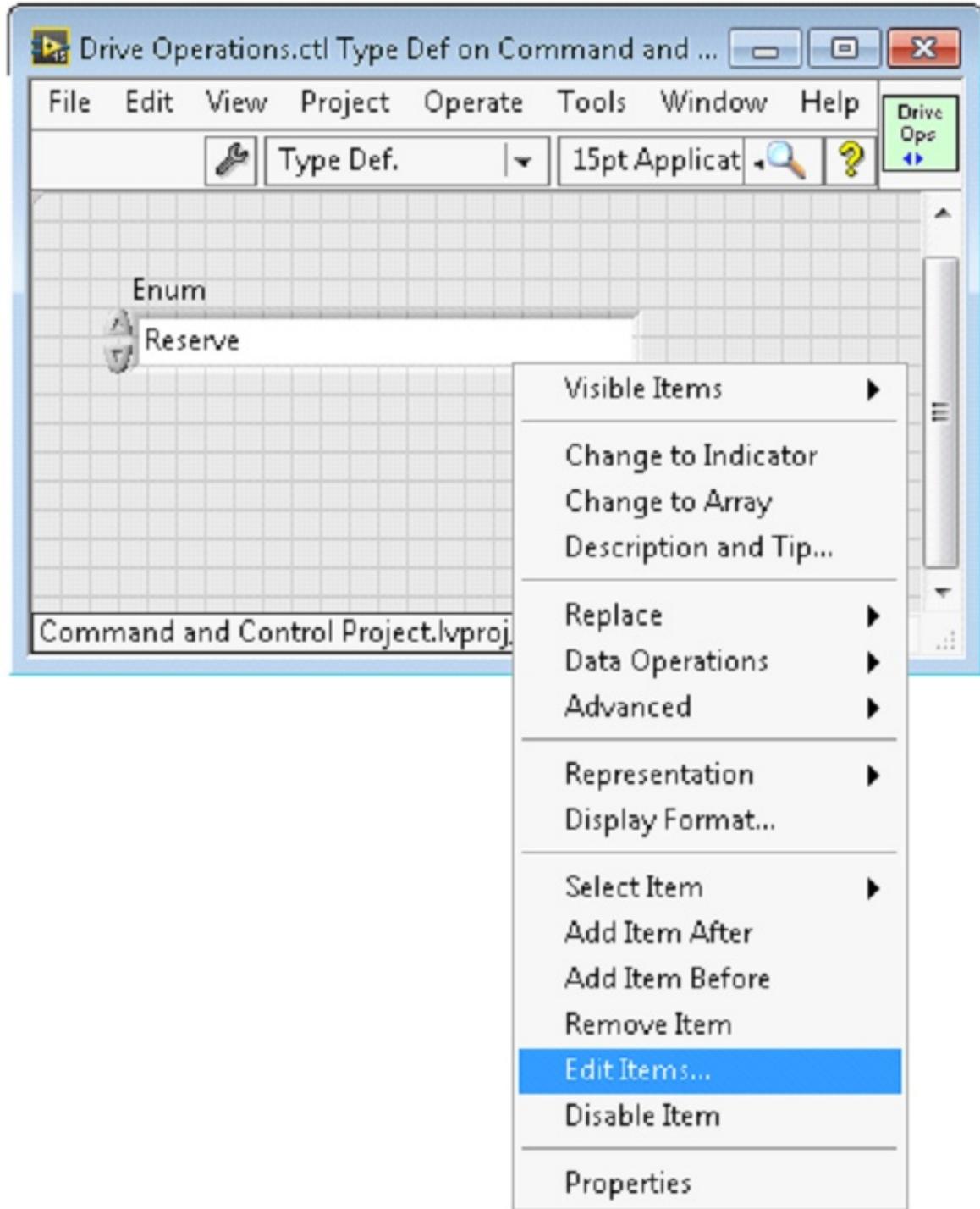
FRC LabVIEW Programming



FRC LabVIEW Programming

First, save the new VI with a descriptive name such as “Drive for Distance”. Next, determine whether the new command needs a new value added the Drive Operations enum typedef. The initial project code already has an enum value of Drive for Distance, but the following image shows how you would add one if needed.

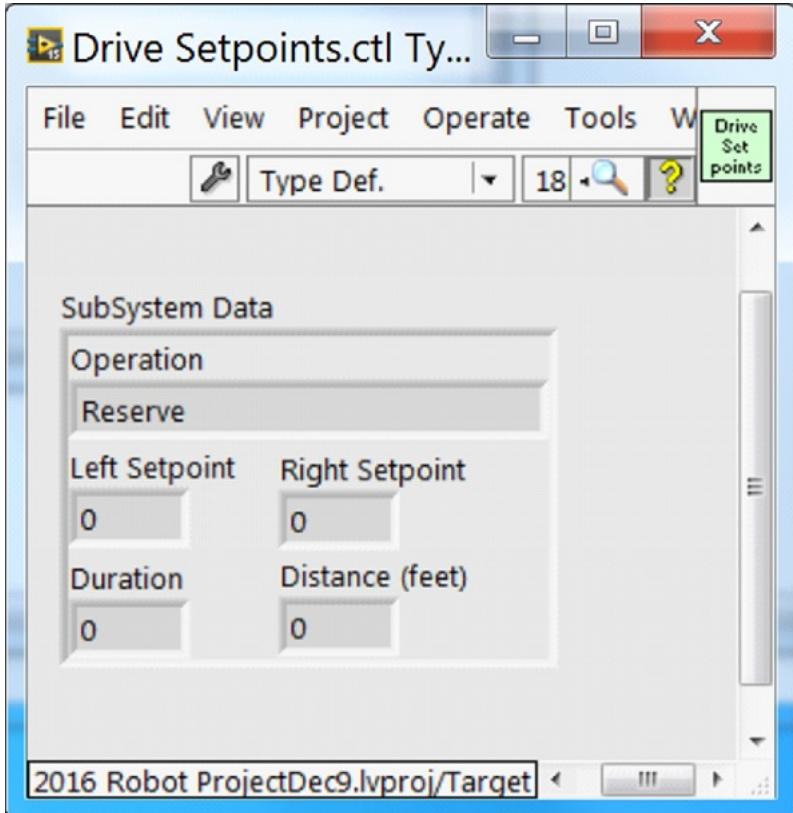
FRC LabVIEW Programming



If a command needs additional information to execute, add it to the setpoints control. By default, the Drive subsystem has fields for the Left Setpoint, Right Setpoint, and Duration along with the

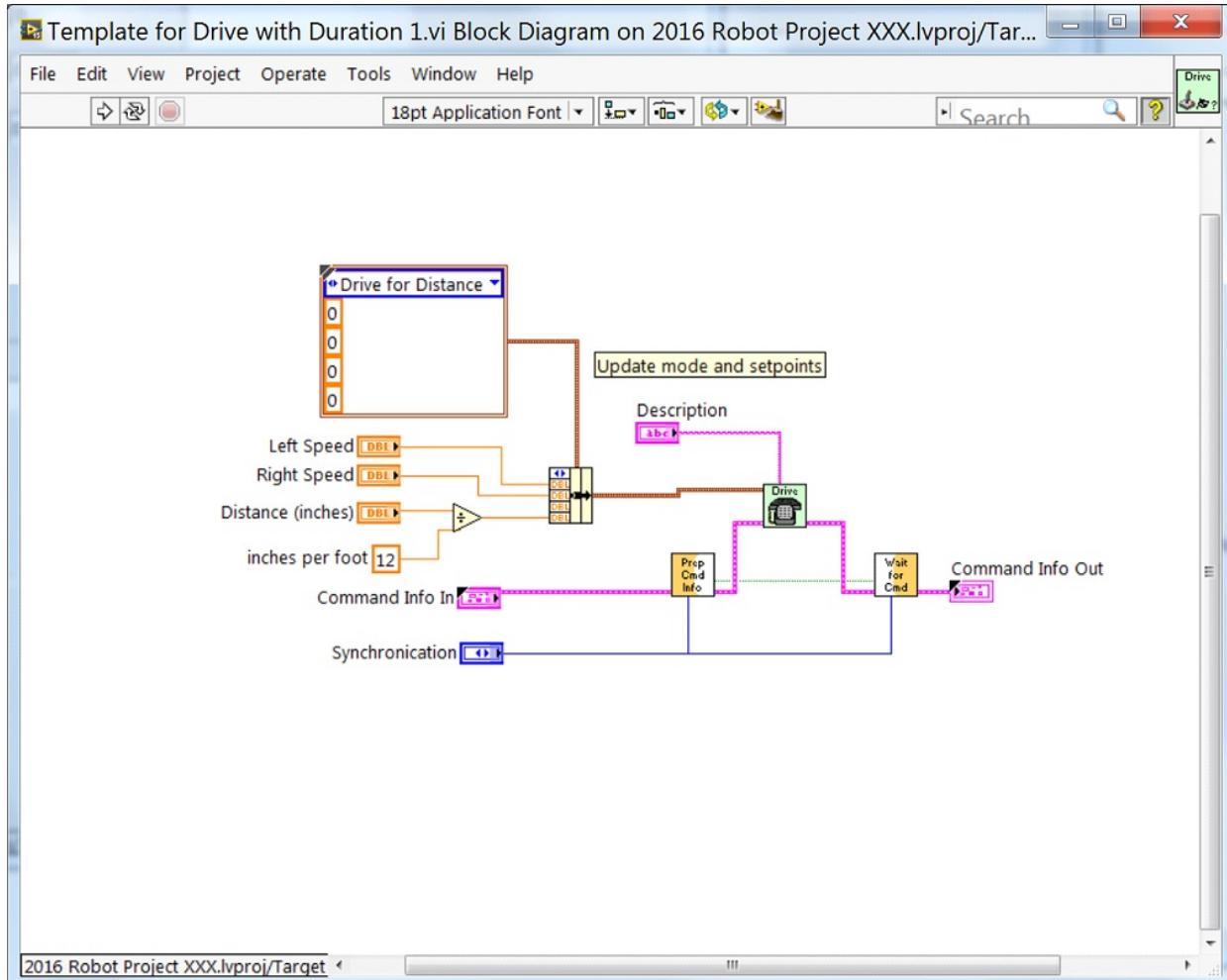
FRC LabVIEW Programming

operation to be executed. The Drive for Distance command could reuse Duration as distance, but let's go ahead and add a numeric control to the Drive Setpoints.ctl called Distance (feet).



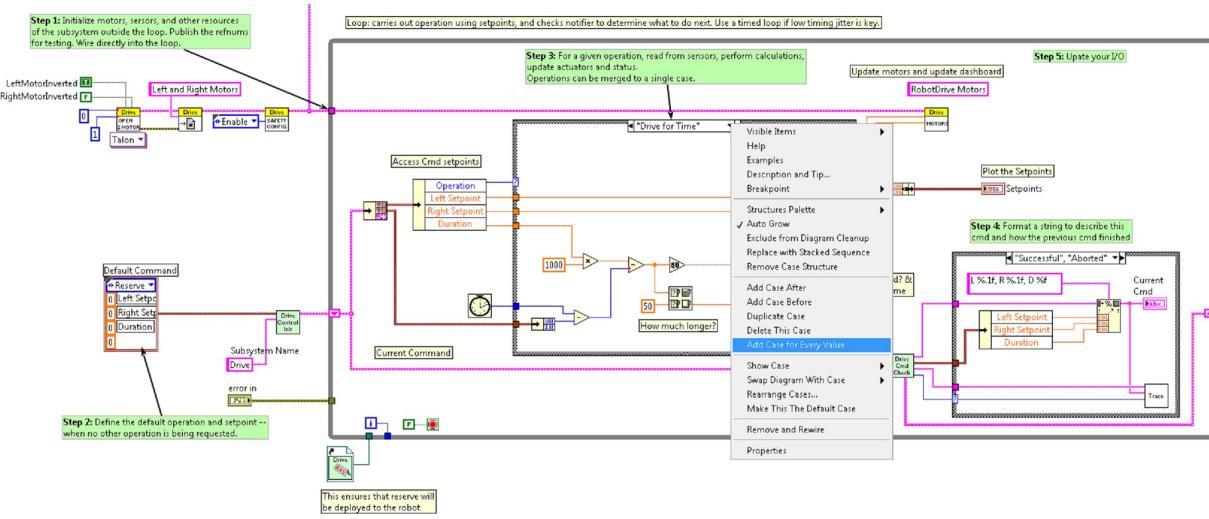
Once we have all of the fields needed to specify our command, we can modify the newly created Drive for Distance.vi. As shown below, select Drive for Distance from the enum's drop down menu and add a VI parameters to specify distance, speeds, etc. If the units do not match, the command VI is a great place to map between units.

FRC LabVIEW Programming

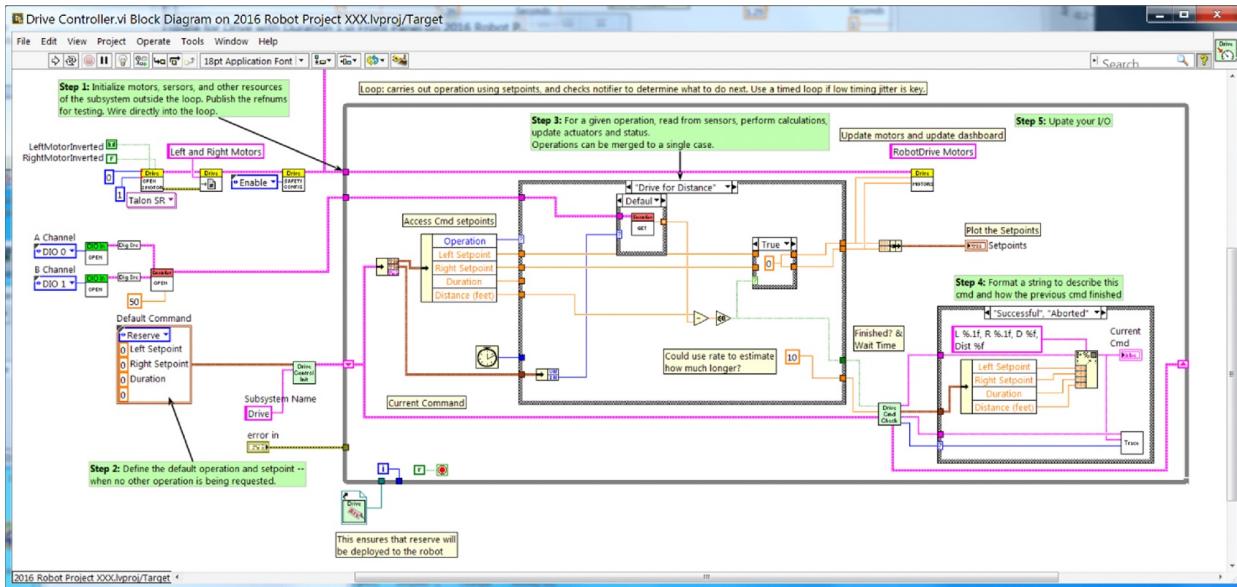


Next, add code to the Drive Controller to define what happens when the Drive for Distance command executes. Right click on the Case Structure and Duplicate or Add Case for Every Value. This will create a new “Drive for Distance” case.

FRC LabVIEW Programming



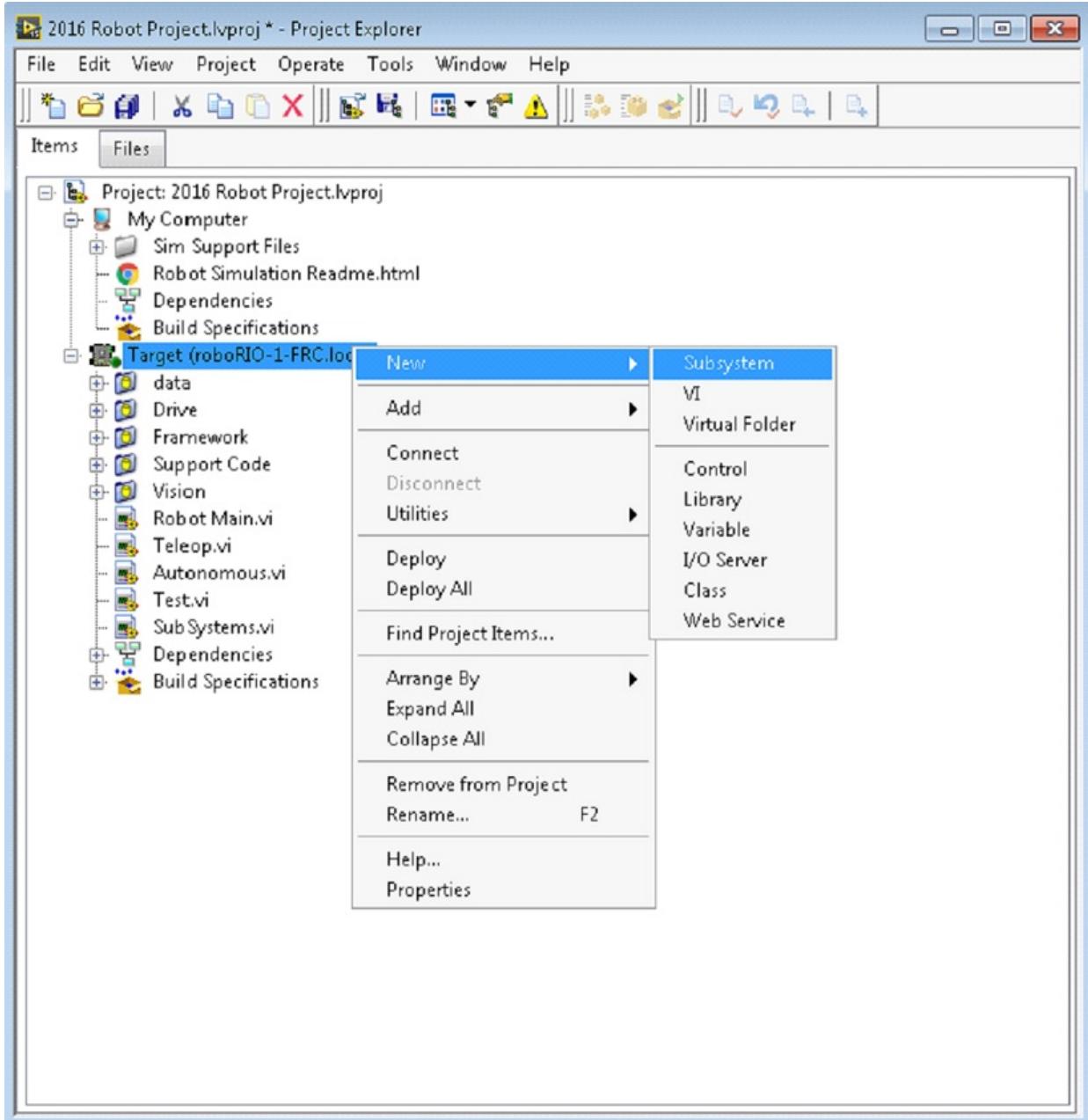
In order to access new setpoint fields, grow the “Access Cmd setpoints” unbundle node. Open your encoder(s) on the outside, to the left of the loop. In the new diagram of the case structure, we added a call to reset the encoder on the first loop iteration and read it otherwise. There is also some simple code that compares encoder values and updates the motor power. If new controls are added to the setpoints cluster, you should also consider adding them to the Trace. The necessary changes are shown in the image below.



Part 5: Creating a Subsystem

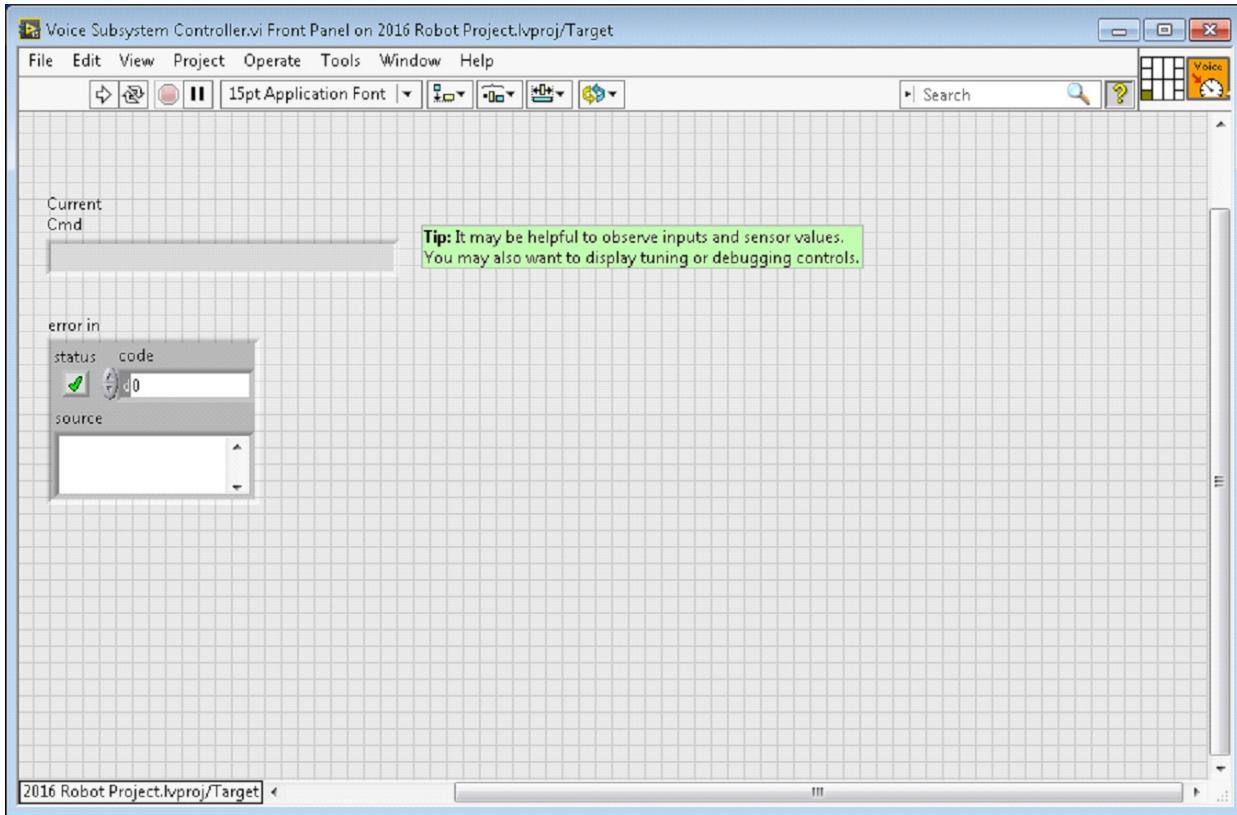
In order to create a new subsystem, right click on the roboRIO target and select New» Subsystem. In the pop up dialog box, enter the name of the subsystem, list the operational modes, and specify the color of the icon.

FRC LabVIEW Programming

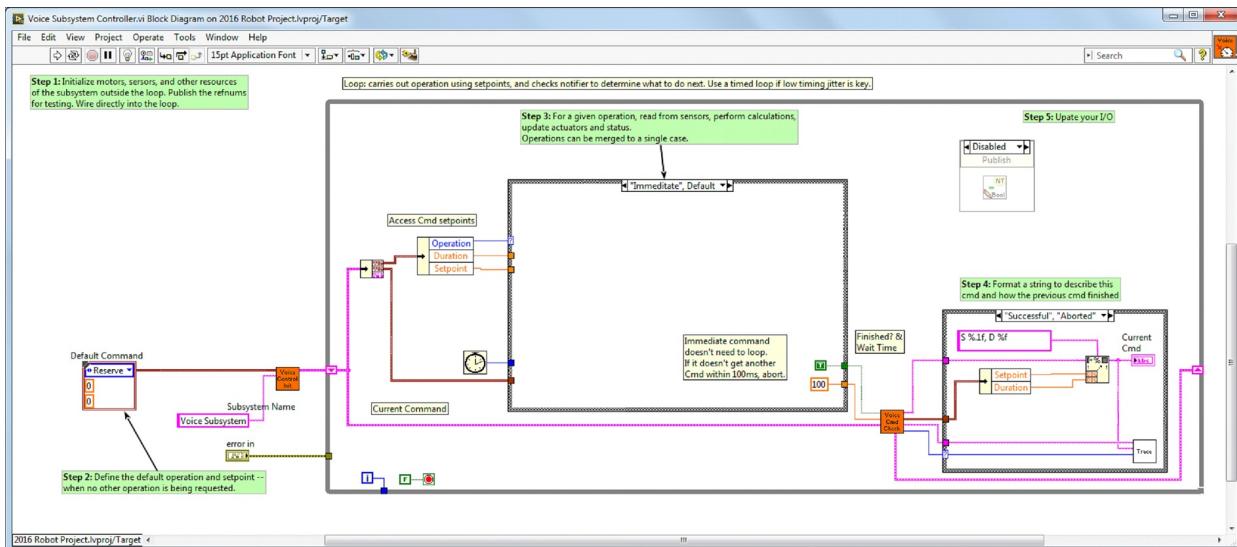


When you click OK, the subsystem folder will be generated and added to the project disk folder and tree. It will contain a base implementation of the VIs and controls that make up a subsystem. A call to the new controller will be inserted into the Subsystems VI. The controller VI will open, ready for you to add I/O and implement state machine or control code. Generated VI icons will use the color and name provided in the dialog. The generated code will use typedefs for set point fields and operations.

FRC LabVIEW Programming



Below is the block diagram of the newly created subsystem. This code will be generated automatically when you create the subsystem.



LabVIEW Resources

LabVIEW Resources



To learn more about programming in LabVIEW and specifically programming FRC robots in LabVIEW, check out the following resources.

LabVIEW Basics

National Instruments provides a [combination of videos](#) and [traditional text/picture tutorials on the basics of LabVIEW](#). These tutorials can help you get acquainted with the LabVIEW environment and the basics of the graphical, dataflow programming model used in LabVIEW.

NI FRC Tutorials

National Instruments also hosts many [FRC specific tutorials and presentations ranging from basic to advanced](#). For an in-depth single resource check out the FRC Basic and Advanced Training Classes linked near the bottom of the page.

Installed Tutorials and Examples

There are also tutorials and examples for all sorts of tasks and components provided as part of your LabVIEW installation. To access the tutorials, from the LabVIEW Splash screen (the screen that appears when the program is first launched) click on the Tutorials tab on the left side. Note that the tutorials are all in one document, so once it is open you are free to browse to other tutorials without returning to the splash screen.

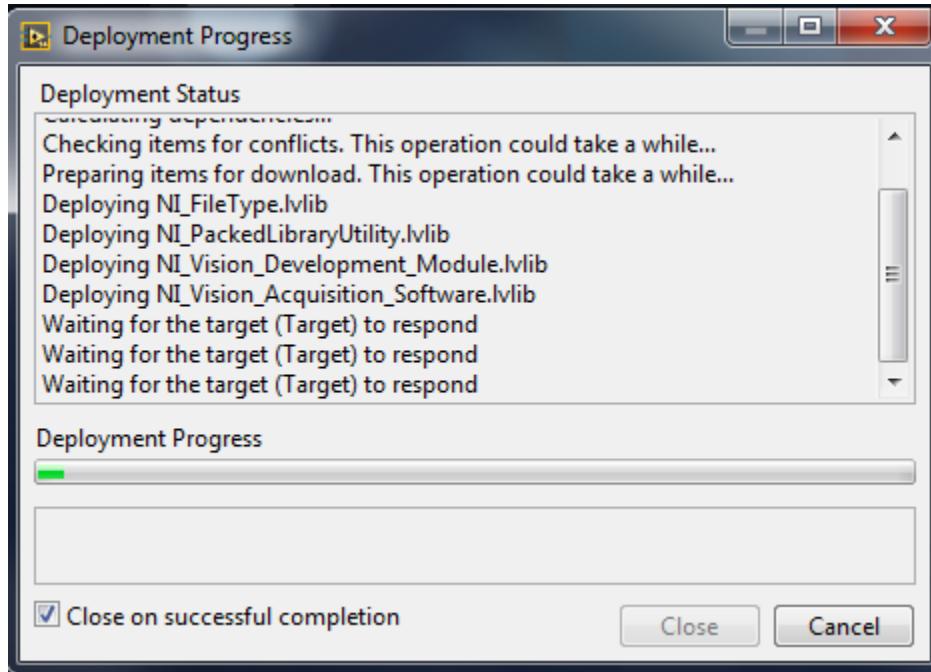
To access the examples either click the Support tab, then Find FRC Examples or anytime you're working on a program open the Help menu, select Find Examples and open the FRC Robotics folder.

Waiting for Target to Respond - Recovering from bad loops



If you download LabVIEW code which contains an unconstrained loop (a loop with no delay) it is possible to get the roboRIO into a state where LabVIEW is unable to connect to download new code. This document explains the process required to load new, fixed, code to recover from this state.

The Symptom

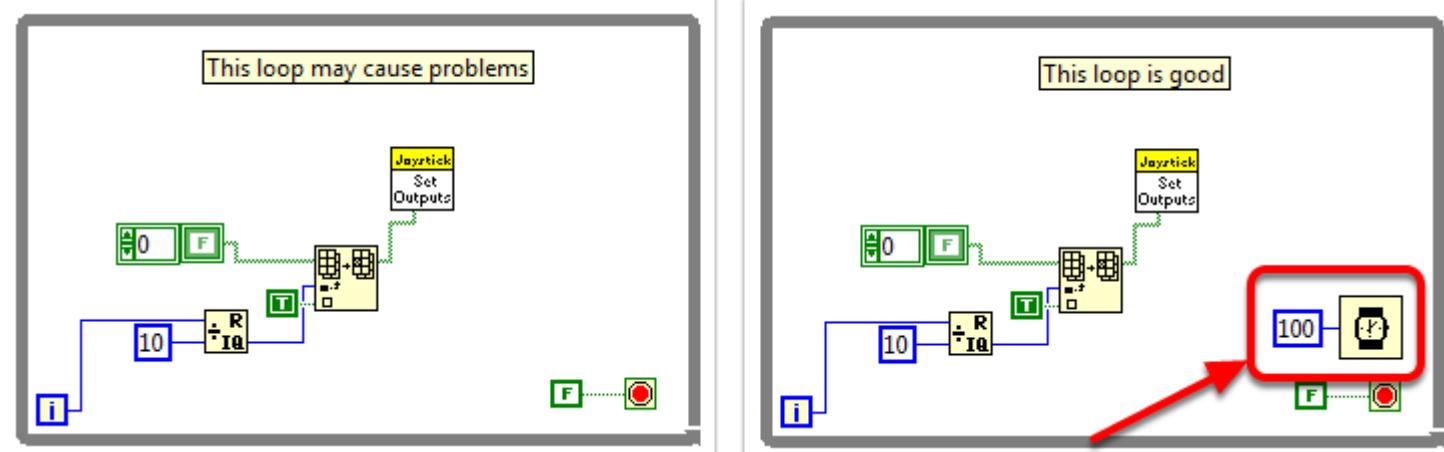


FRC LabVIEW Programming

The primary symptom of this issue is attempts to download new robot code hang at the "Waiting for the target (Target) to respond" step as shown above. Note that there are other possible causes of this symptom (such as switching from a C++\Java program to LabVIEW program) but the steps described here should resolve most or all of them.

Click Cancel to close the download dialog.

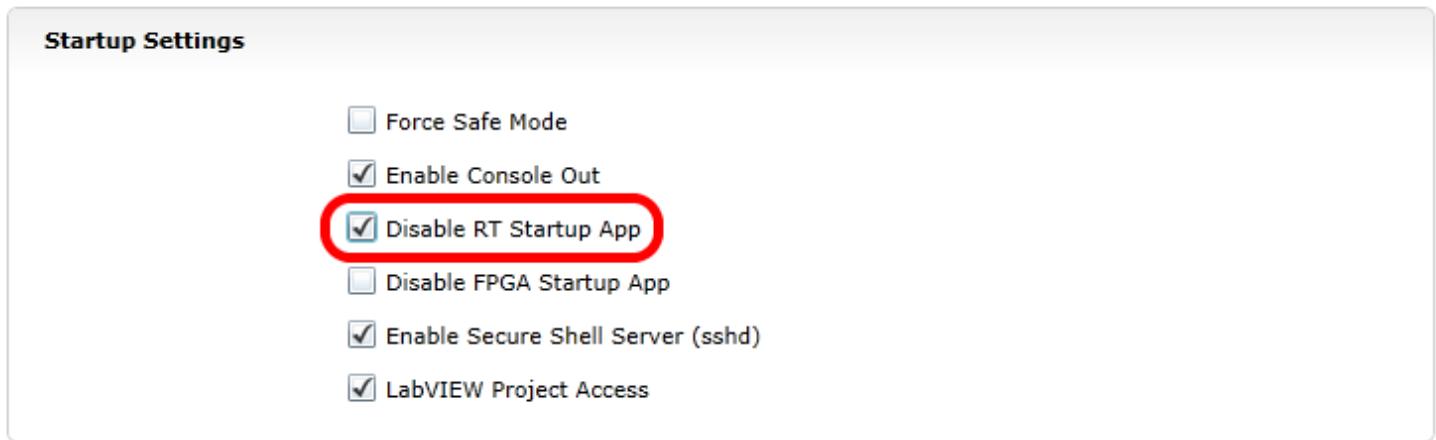
The Problem



One common source of this issue is unconstrained loops in your LabVIEW code. An unconstrained loop is a loop which does not contain any delay element (such as the one on the left). If you are unsure where to begin looking, Disabled.VI, Periodic Tasks.VI and Vision Processing.VI are the common locations for this type of loop. To fix the issue with the code, add a delay element such as the Wait (ms) VI from the Timing palette, found in the right loop.

FRC LabVIEW Programming

Set No App



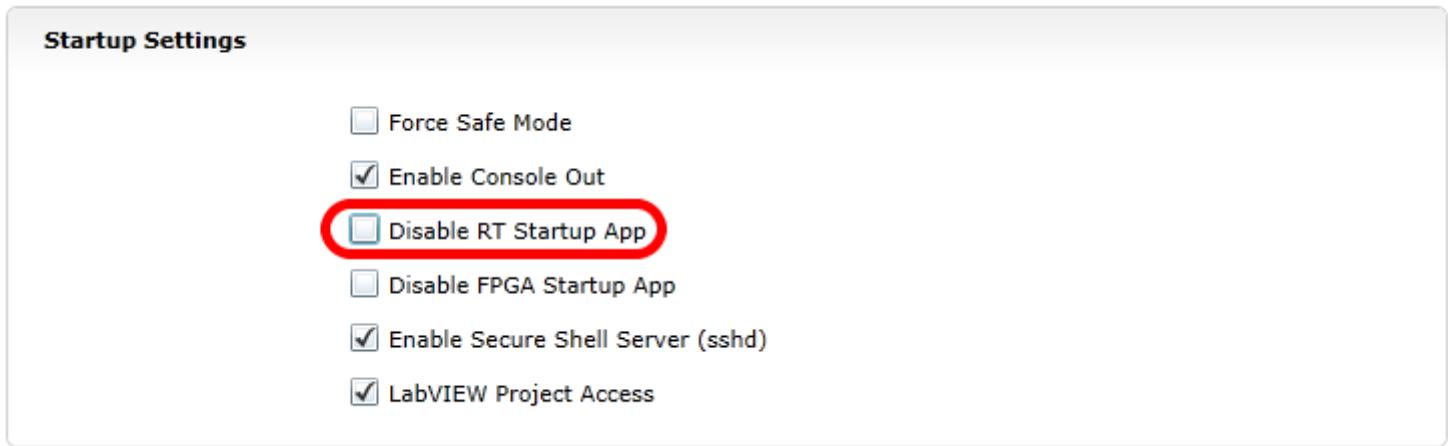
Using the roboRIO webserver (see the article [RoboRIO Webdashboard](#) for more details). Check the box to "Disable RT Startup App".

Reboot

Reboot the roboRIO, either using the Reset button on the device or by click Restart in the top right corner of the webpage.

FRC LabVIEW Programming

Clear No App



Using the roboRIO webserver (see the article [RoboRIO Webdashboard](#) for more details). Uncheck the box to "Disable RT Startup App".

Load LabVIEW Code

Load LabVIEW code (either using the Run button or Run as Startup). Make sure to set LabVIEW code to Run as Startup before rebooting the roboRIO or you will need to follow the instructions above again.

Talon SRX CAN

The Talon SRX motor controller is a CAN-enabled "smart motor controller" from Cross The Road Electronics/VEX Robotics. The Talon SRX can be controlled over the CAN bus or PWM interface. When using the CAN bus control, this device can take inputs from limit switches and potentiometers, encoders, or similar sensors in order to perform advanced control such as limiting or PID(F) closed loop control on the device.

Extensive documentation about programming the Talon SRX in all three FRC languages can be found in the [Talon SRX Software Reference Manual](#) on CTRE's [Talon SRX product page](#).

Note: CAN Talon SRX has been removed from WPILib. See [this blog](#) for more info and find the CTRE Toolsuite installer here: http://www.ctr-electronics.com/control-system/hro.html#product_tabs_technical_resources

How To Toggle Between Two Camera Modes



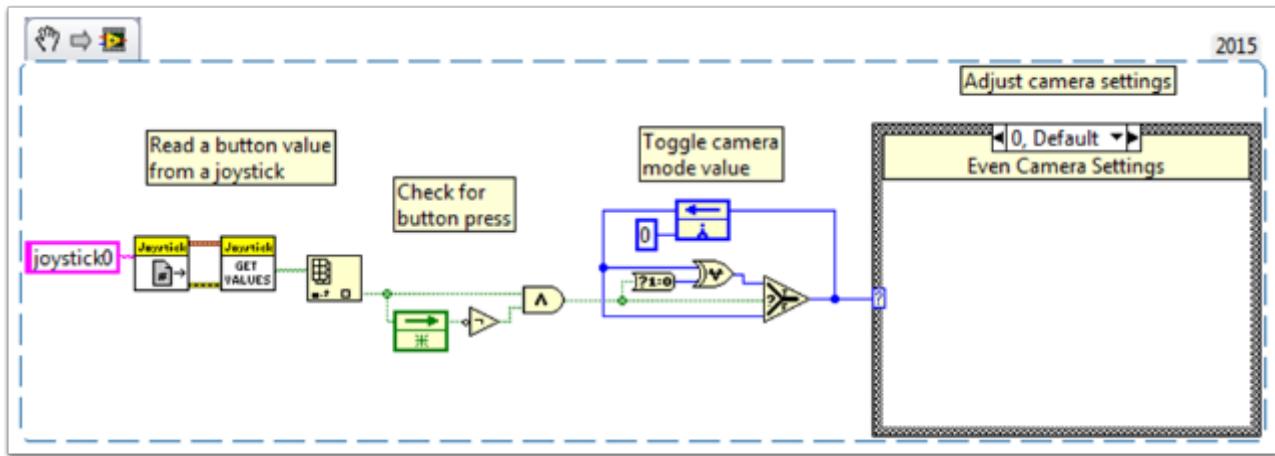
This code shows how to use a button to toggle between two distinct camera modes. The code consists of four stages.

In the first stage, the value of a button on the joystick is read.

Next, the current reading is compared to the previous reading using a Feedback Node and some Boolean arithmetic. Together, these ensure that the camera mode is only toggled when the button is initially pressed rather than toggling back and forth multiple times while the button is held down.

After that, the camera mode is toggled by masking the result of the second stage over the current camera mode value. This is called bit masking and by doing it with the XOR function the code will toggle the camera mode when the second stage returns true and do nothing otherwise.

Finally, you can insert the code for each camera mode in the case structure at the end. Each time the code is run, this section will run the code for the current camera mode.



LabVIEW Examples and Tutorials



Popular Tutorials

[Autonomous Timed Movement Tutorial](#)

- Move your robot autonomously based on different time intervals
- [See more on Autonomous Movement](#)

[Basic Motor Control Tutorial](#)

- Setup your roboRIO motor hardware and software
- Learn to setup the FRC Control System and FRC Robot Project
- [See more on Motor Control](#)

[Image Processing Tutorial](#)

- Learn basic Image Processing techniques and how to use NI Vision Assistant
- [See more on Cameras and Image Processing](#)

[PID Control Tutorial](#)

- What is PID Control and how can I implement it?

[Command and Control Tutorial](#)

- What is Command and Control?
- How do I implement it?

[Driver Station Tutorial](#)

- Get to know the FRC Driver Station

[Test Mode Tutorial](#)

FRC LabVIEW Programming

- Learn to setup and use Test Mode

Looking for more examples and discussions? Search through more documents or post your own discussion, example code, or tutorial by [clicking here!](#) Don't forget to mark your posts with a tag!

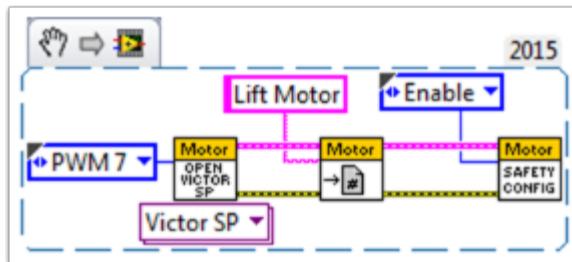
Add an Independent Motor to a Project



Once your drive that controls the wheels is all set, you might need to add an additional motor to control something completely independent of the wheels, such as an arm. Since this motor will not be part of your tank, arcade, or mecanum drive, you'll definitely want independent control of it.

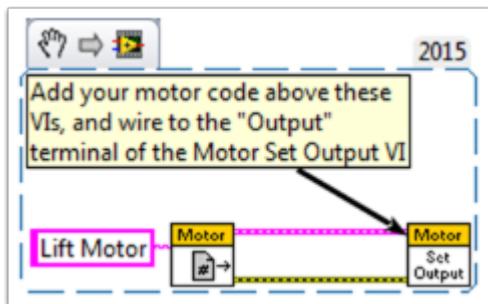
These VI Snippets show how to set up a single motor in a project that may already contain a multi-motor drive. If you see the HAND>ARROW>LABVIEW symbol, just drag the image into your block diagram, and voila: code! Ok, here's how you do it.

FIRST, create a motor reference in the Begin.vi, using the Motor Control Open VI and Motor Control Refnum Registry Set VI. These can be found by right-clicking in the block diagram and going to WPI Robotics Library>>RobotDrive>>Motor Control. Choose your PWM line and name your motor. I named mine "Lift Motor" and connected it to PWM 7. (I also included and enabled the Motor Control Safety Config VI, which automatically turns off the motor if it loses connection.)

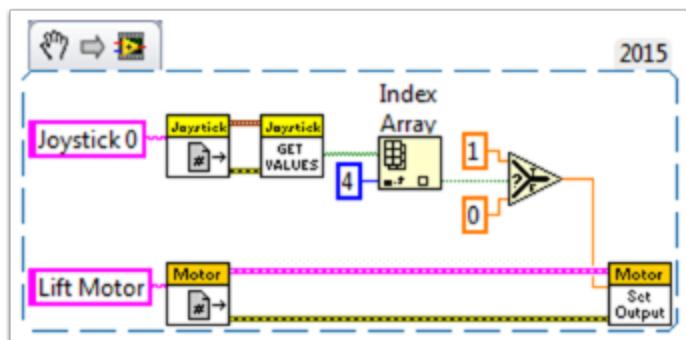


Now, reference your motor (the name has to be exact) in the Teleop.vi using the Motor Control Refnum Registry Get VI and tell it what to do with the Motor Control Set Output VI. These are in the same place as the above VIs.

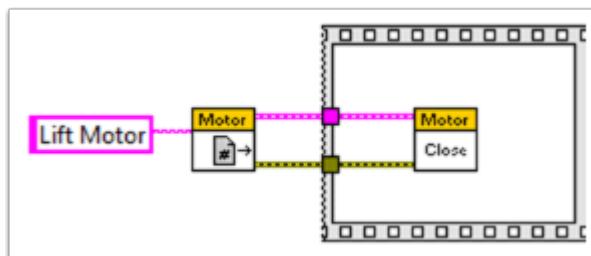
FRC LabVIEW Programming



For example, the next snippet tells the Lift Motor to move forward if button 4 is pressed on Joystick 0 and to remain motionless otherwise. For me, button 4 is the left bumper on my Xbox style controller ("Joystick 0"). For much more in-depth joystick button options, check out [How to Use Joystick Buttons to Control Motors or Solenoids](#).



Finally, we need to close the references in the Finish.vi (just like we do with the drive and joystick), using the **Motor Control Refnum Registry Get VI** and **Motor Control Close VI**. While this picture shows the Close VI in a flat sequence structure by itself, we really want all of the Close VIs in the same frame. You can just put these two VIs below the other Get VIs and Close VIs (for the joystick and drive).



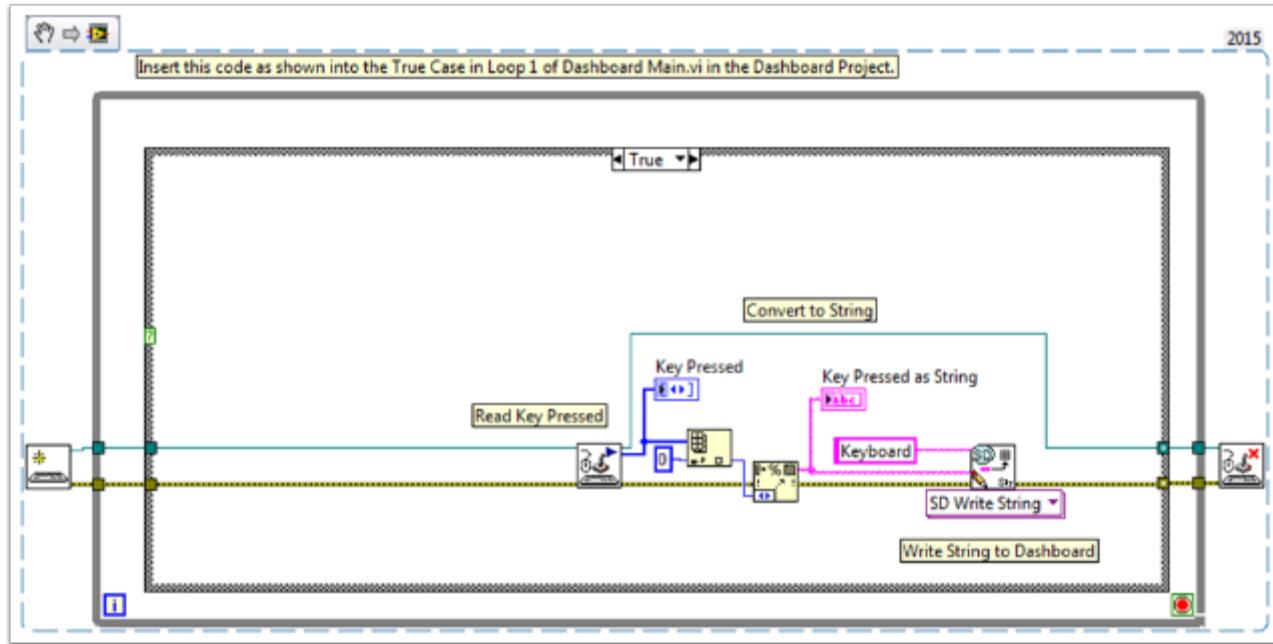
I hope this helps you program the best robot ever! Good luck!

Keyboard Navigation with the roboRIO



This example provides some suggestions for controlling the robot using keyboard navigation in place of a joystick or other controller. In this case, we use the A, W, S, and D keys to control two drive motors in a tank drive configuration.

The first VI Snippet is the code that will need to be included in the Dashboard Main VI. You can insert this code into the True case of Loop 1. The code opens a connection to the keyboard before the loop begins, and on each iteration it reads the pressed key. This information is converted to a string, which is then passed to the Teleop VI in the robot project. When Loop 1 stops running, the connection to the keyboard is closed.



The second VI Snippet is code that should be included in the Teleop VI. This reads the string value from the Dashboard that indicates which key was pressed. A Case Structure then determines which values should be written to the left and right motors, depending on the key. In this case, W is forward, A is left, D is right, and S is reverse. Each case in this example runs the motors at half speed. You can keep this the same in your code, change the values, or add additional code to allow the driver to adjust the speed, so you can drive fast or slow as necessary. Once the motor

FRC LabVIEW Programming

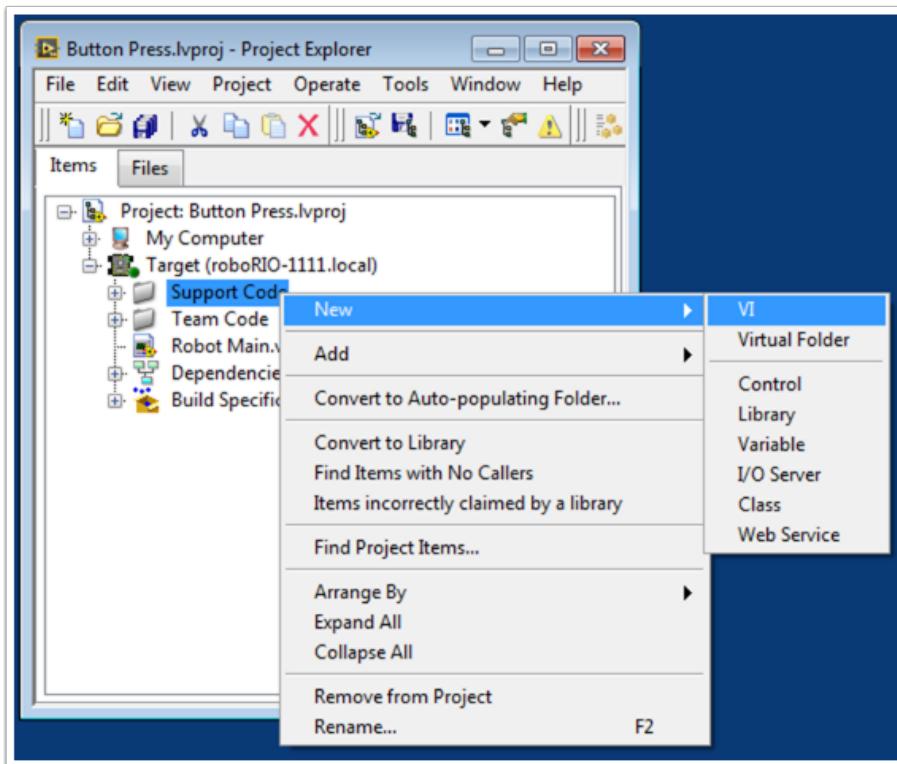
values are selected, they are written to the drive motors, and motor values are published to the dashboard.

Making a One-Shot Button Press



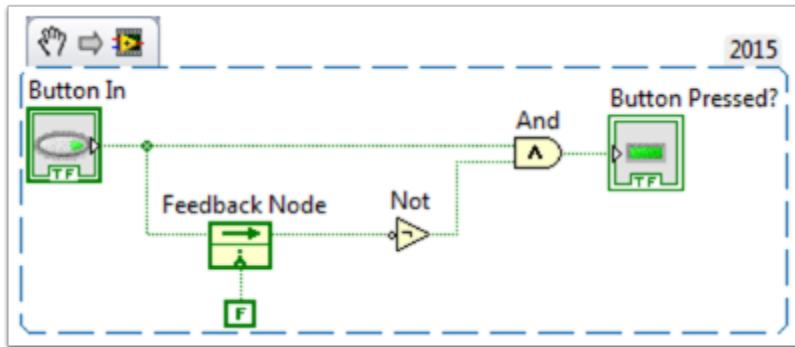
When using the Joystick Get Values function, pushing a joystick button will cause the button to read TRUE until the button is released. This means that you will most likely read multiple TRUE values for each press. What if you want to read only one TRUE value each time the button is pressed? This is often called a "One-Shot Button". The following tutorial will show you how to create a subVI that you can drop into your Teleop.vi to do this.

FIRSTly, create a new VI in the Support Code folder of your project.



Now on the block diagram of the new VI, drop in the following code snippet.

FRC LabVIEW Programming

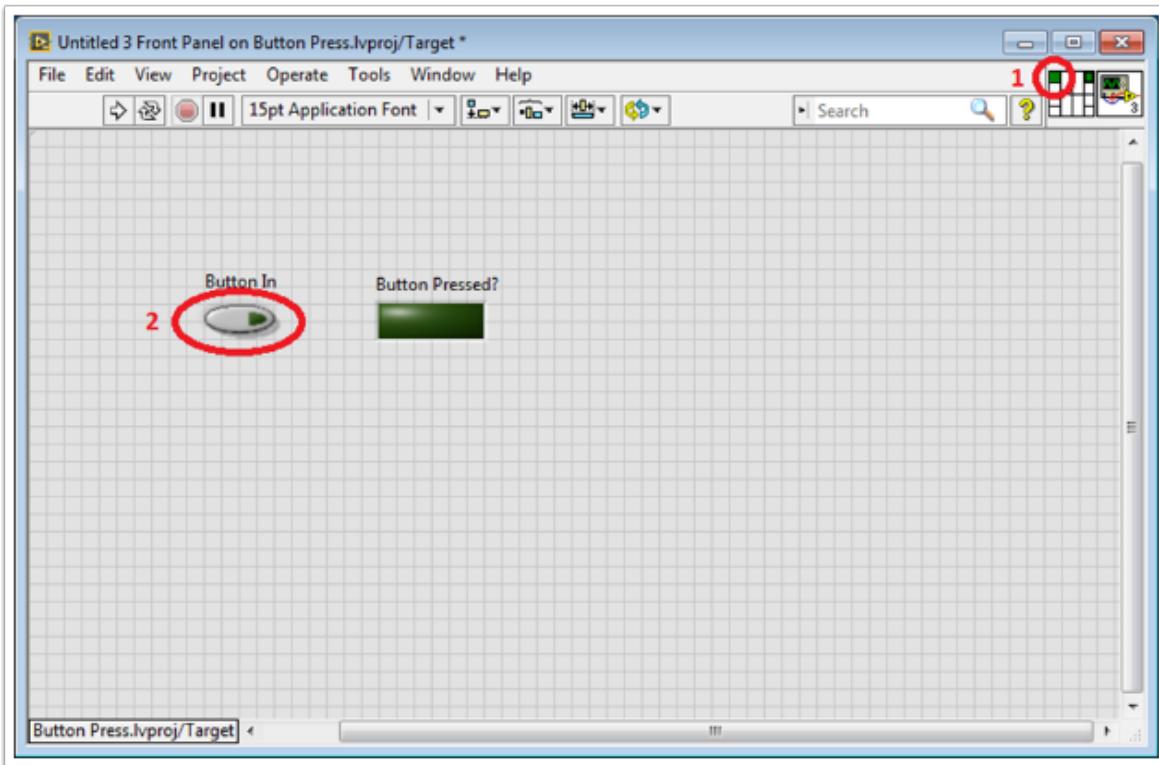


This code uses a function called the Feedback Node. We have wired the current value of the button into the left side of the feedback node. The wire coming out of the arrow of the feedback node represents the previous value of the button. If the arrow on your feedback node is going the opposite direction as shown here, right click to find the option to reverse the direction.

When a button is pressed, the value of the button goes from FALSE to TRUE. We want the output of this VI to be TRUE only when the current value of the button is TRUE, and the previous value of the button is FALSE.

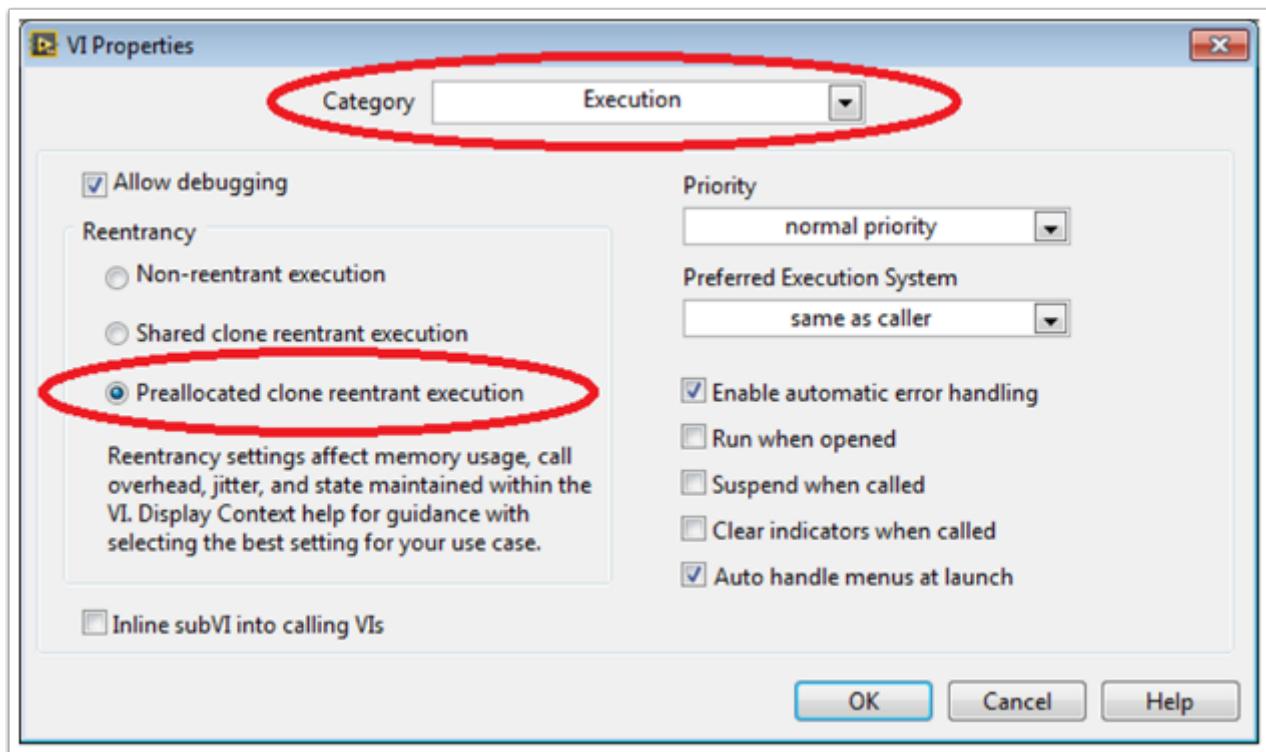
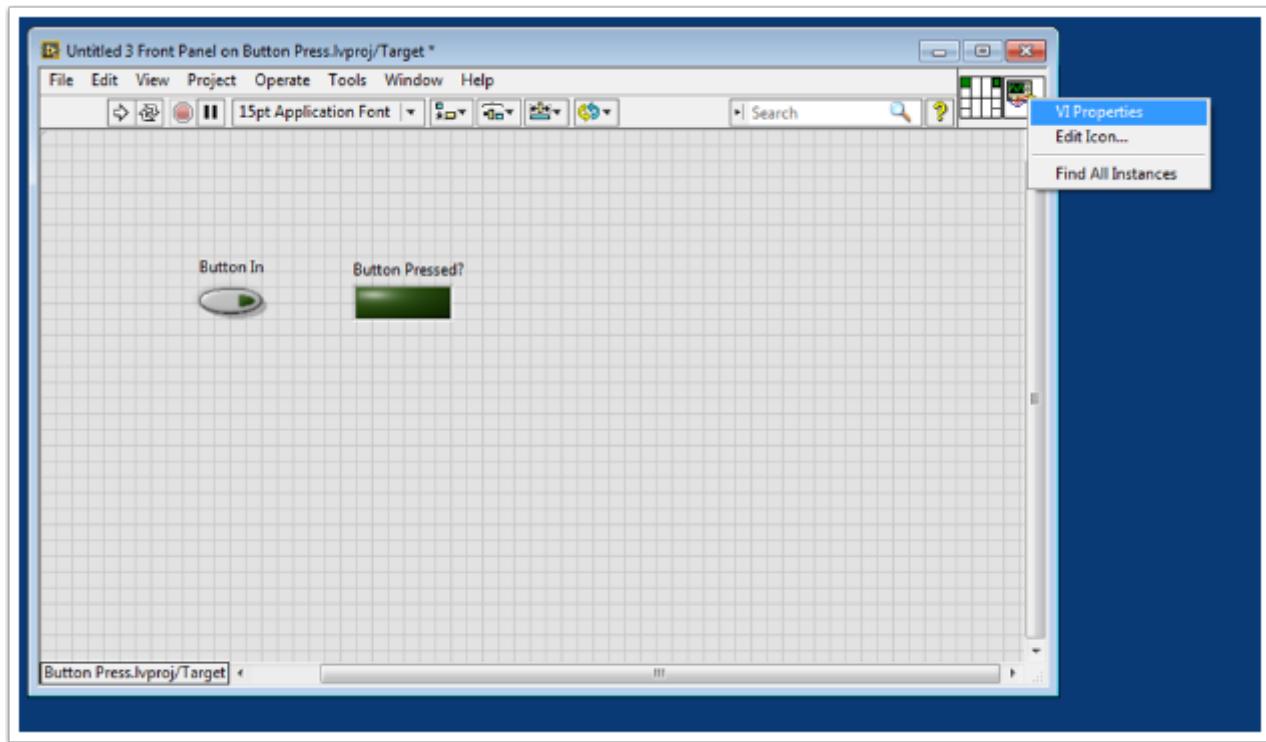
Next we need to connect the boolean control and indicator to the inputs and outputs of the VI. To do this, first click the block on the connector pane, then click the button to connect the two (see the diagram below). Repeat this for the indicator.

FRC LabVIEW Programming



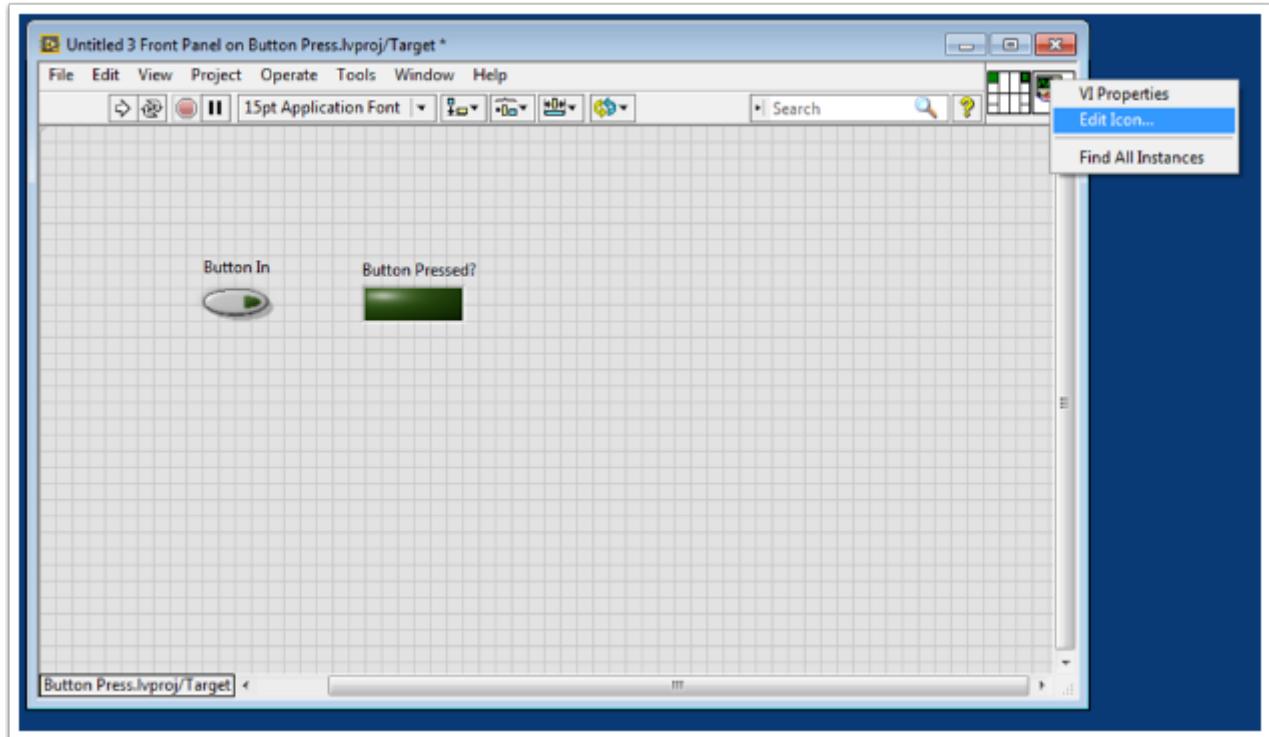
Next, we need to change the properties of this VI so that we can use multiples of this VI in our TeleOp.vi. Right click the VI Icon and go to VI Properties. Then select the category "Execution" and select "Preallocated clone reentrant execution".

FRC LabVIEW Programming



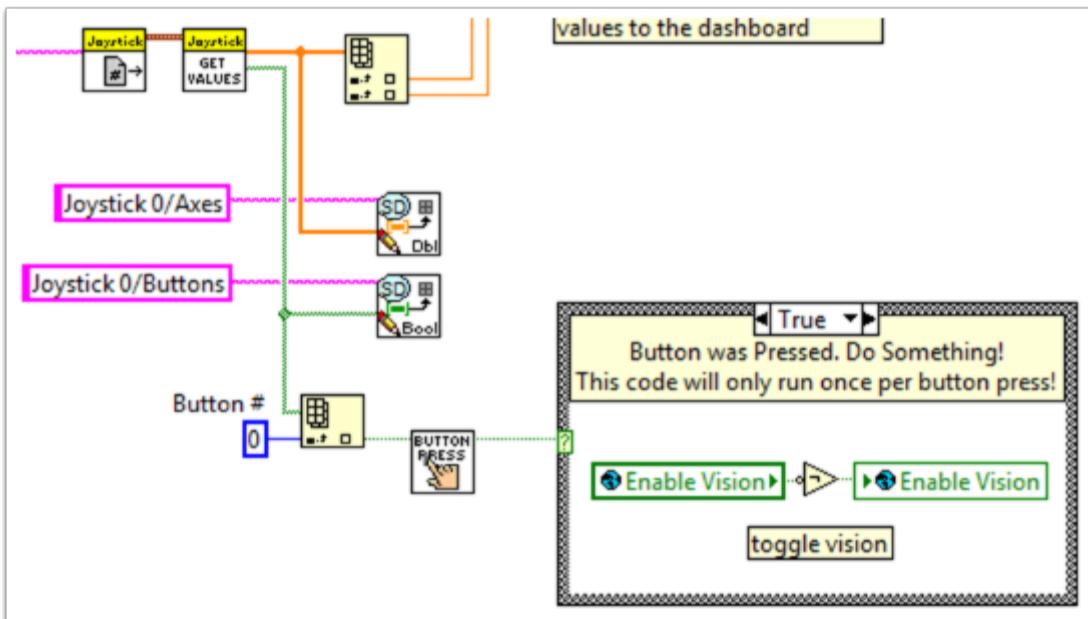
FRC LabVIEW Programming

Lastly, we should change the VI Icon to be more descriptive of the VI's function. Right click the Icon and go to Edit Icon. Create a new Icon.



Finally, save the VI with a descriptive name. You can now drag and drop this VI from the Support Files folder into your TeleOp.vi. Here's an example of how you could use this VI.

FRC LabVIEW Programming



Button_Press.vi

Adding Safety Features to Your Robot Code



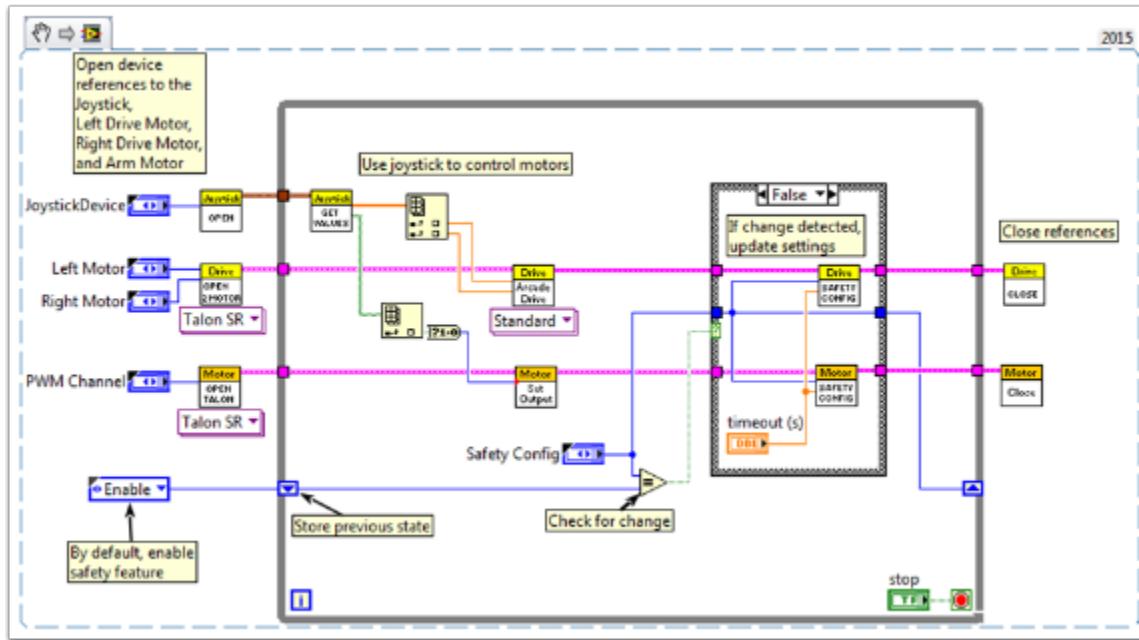
A common problem with complex projects is making sure that all of your code is executing when you expect it to. Problems can arise when tasks with high priority, long execution times, or frequent calls hog processing power on the roboRIO. This leads to what is known as "starvation" for the tasks that are not able to execute due to the processor being busy. In most cases this will simply slow the reaction time to your input from the joysticks and other devices. However, this can also cause the drive motors of your robot to stay on long after you try to stop them. To avoid any robotic catastrophes from this, you can implement safety features that check for task input starvation and automatically shut down potentially harmful operations.

There are built-in functions for the motors that allow easy implementation of safety checks. These functions are:

- Robot Drive Safety Configuration
- Motor Drive Safety Configuration
- Relay Safety Configuration
- PWM Safety Configuration
- Solenoid Safety Configuration
- Robot Drive Delay and Update Safety

In all of the Safety Configuration functions, you can enable and disable the safety checks while your programming is running and configure what timeout you think is appropriate. The functions keep a cache of all devices that have the safety enabled and will check if any of them have exceeded their time limit. If any has, all devices in the cache will be disabled and the robot will come to an immediate stop or have its relay/PWM/solenoid outputs turned off. The code below demonstrates how to use the Drive Safety Configuration functions to set a maximum time limit that the motors will receive no input before being shut off.

FRC LabVIEW Programming



To test the safety shut-off, try adding a Wait function to the loop that is longer than your timeout!

The final function that relates to implementing safety checks--Robot Drive Delay and Update Safety--allows you to put the roboRIO in Autonomous Mode without exceeding the time limit. It will maintain the current motor output without making costly calls to the Drive Output functions, and will also make sure that the safety checks are regularly updated so that the motors will not suddenly stop.

Overall, it is highly recommended that some sort of safety check is implemented in your project to make sure that your robot is not unintentionally left in a dangerous state!

How to Use Joystick Buttons to Control Motors or Solenoids

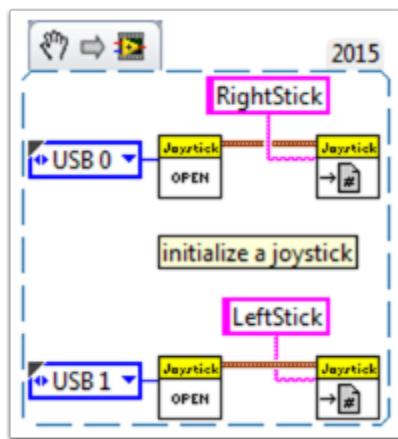


As we all get our drive systems working, we are moving on to connecting our auxiliary devices such as motors and solenoids. With this, we will generally use joystick buttons to control these devices. To get started with this, we'll go through several ways to control devices with joystick buttons.

Did you know that you can click and drag a VI Snippet from a document like this right into your LabVIEW code? Try it with the snippets in this document.

Setup:

No matter what the configuration, you'll need to add one, two, or more (if you're really excited) joysticks to the "Begin.vi". The first example uses 2 joysticks and the others only use one. Give each one a unique name so we can use it in other places, like the snippet below. I named them "LeftStick" and "RightStick" because they are on the left and right sides of my desk. If your joysticks are already configured, great! You can skip this step.

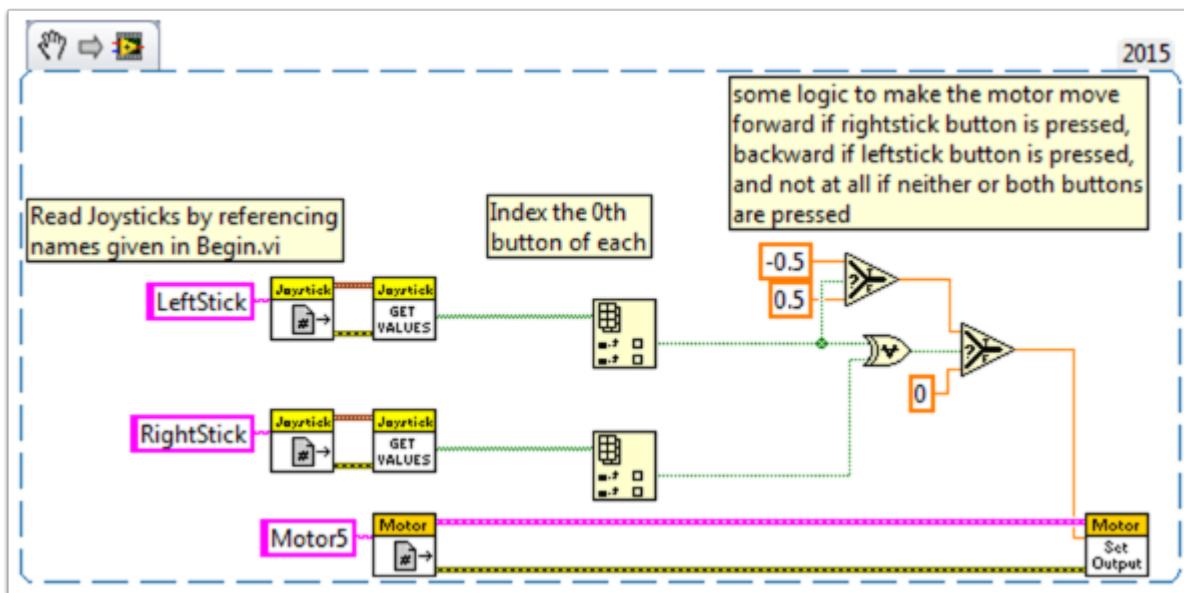


The rest of the code in this document will be placed in the "Teleop.VI" This is where we will be programming our joystick buttons to control different aspects of our motors or solenoids.

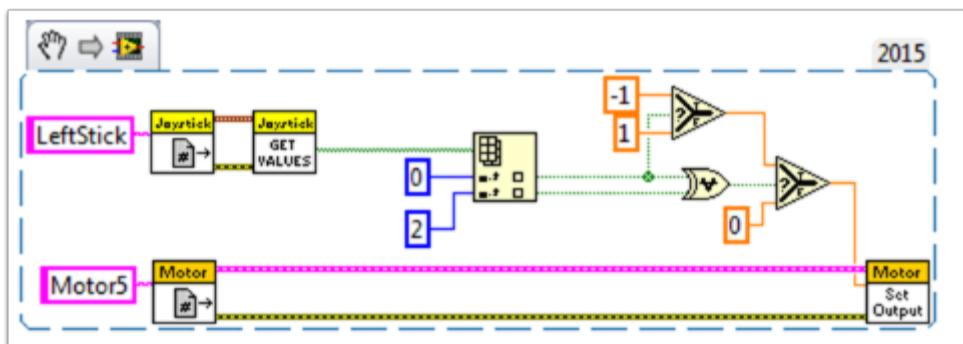
FRC LabVIEW Programming

Scenario 1: I want a motor to move one way when I press one button and the other way when I press a different button.

This code uses button 0 on two different joysticks to control the same motor. If button 0 on LeftStick is pressed, the motor moves backward, and if button 0 on RightStick is pressed, the motor moves forward. If both buttons are pressed or neither button is pressed, the motor doesn't move. Here I named my motor reference "Motor5", but you can name your motor whatever you want in the "Begin.vi"



You may want to use multiple buttons from the same joystick for control. For an example of this, look at the following VI snippet or the VI snippet in Scenario 2.



Here I used joystick buttons 0 and 2, but feel free to use whatever buttons you need.

FRC LabVIEW Programming

Scenario 2: I want different joystick buttons move at various speeds.

This example could be helpful if you need to have one motor do different things based on the buttons you press. For instance, let's say my joystick has a trigger (button 0) and 4 buttons on top (buttons 1 through 4). In this case, the following buttons should have the following functions:

- button 1 - move backward at half speed
- button 2 - move forward at half speed
- button 3 - move backward at 1/4 speed
- button 4 - move forward at 1/4 speed
- trigger - full speed ahead! (forward at full speed)

We would then take the boolean array from the "JoystickGetValues.vi" and wire it to a "Boolean Array to Number" node (Numeric Palette-Conversion Pallette). This converts the boolean array to a number that we can use. Wire this numeric to a case structure.

Each case corresponds to a binary representation of the values in the array. In this example, each case corresponds to a one-button combination. We added six cases: 0 (all buttons off), 1 (button 0 on), 2 (button 1 on), 4 (button 2 on), 8 (button 3 on), and 16 (button 4 on). Notice we skipped value 3. 3 would correspond to buttons 0 and 1 pressed at the same time. We did not define this in our requirements so we'll let the default case handle it.

It might be helpful to review the LabVIEW 2014 Case Structure Help document here:

http://zone.ni.com/reference/en-XX/help/371361L-01/glang/case_structure/

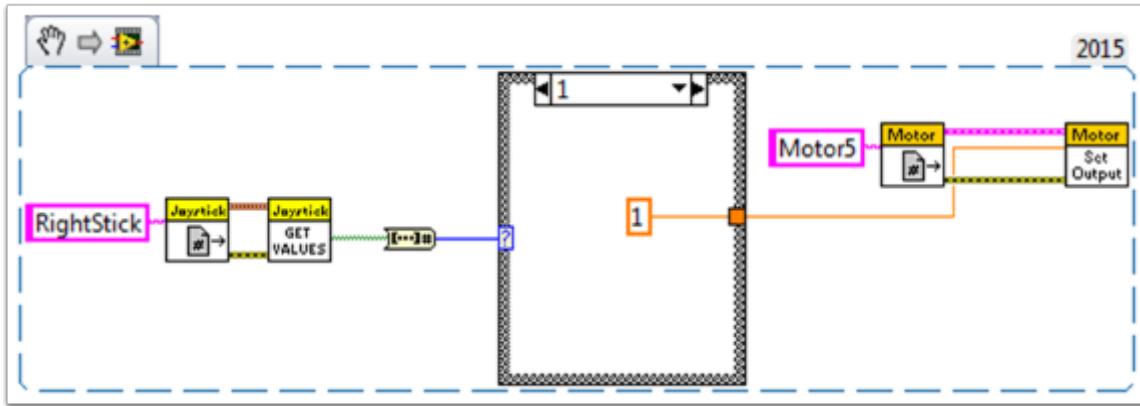
There are also 3 Community Tutorials on case structures here:

<https://decibel.ni.com/content/docs/DOC-11621>

<https://decibel.ni.com/content/docs/DOC-11622>

<https://decibel.ni.com/content/docs/DOC-11623>

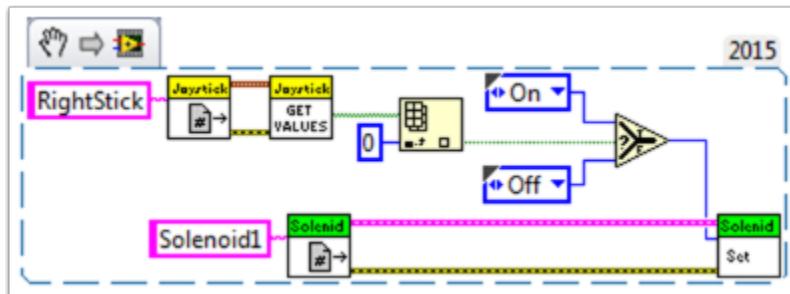
FRC LabVIEW Programming



Since our requirements were simple, we only need a single constant in each case. For case 1 (full ahead) we use a 1, for case 2 (half back) we use a -0.5, etc. We can use any constant value between 1 and -1. I left case 0 as the default so if multiple buttons are pressed (any undefined state was reached) the motor will stop. You of course are free to customize these states however you want.

Scenario 3: I want to control a solenoid with my joystick buttons.

By now, we are familiar with how the joystick outputs the buttons in an array of booleans. We need to index this array to get the button we are interested in, and wire this boolean to a select node. Since the "Solenoid Set.vi" requires a Enum as an input, the easiest way to get the enum is to right click the "Value" input of the "Solenoid Set.vi" and select "Create Constant". Duplicate this constant and wire one copy to the True terminal and one to the False terminal of the select node. Then wire the output of the select node to the "Value" input of the solenoid VI.



Hope this helps! Feel free to comment below.

Happy Roboting!

Local and Global Variables in LabVIEW for FRC



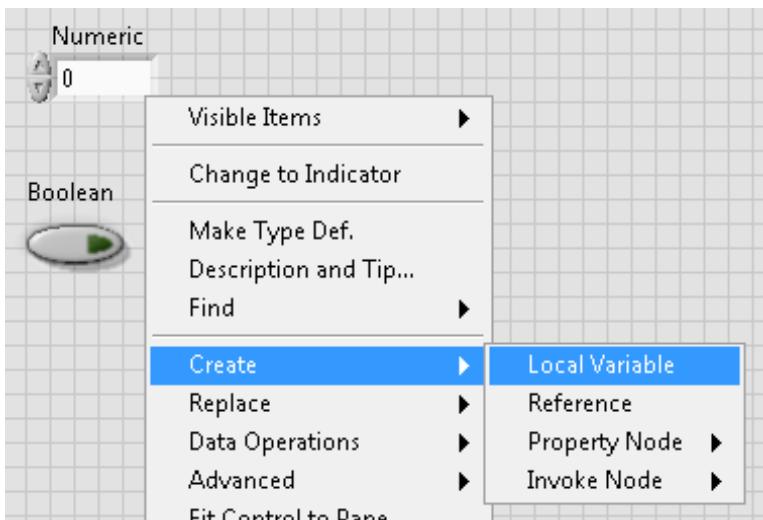
This example serves as an introduction to local and global variables, how they are used in the default LabVIEW for FRC Robot Project, and how you might want to use them in your project.

Local variables and global variables may be used to transfer data between locations within the same VI (local variables) or within different VI's (global variables), breaking the conventional [Data Flow Paradigm](#) for which LabVIEW is famous. Thus, they may be useful when, for whatever reason, you cannot wire the value directly to the node to another.

Note: One possible reason may be that you need to pass data between consecutive loop iterations; Miro_T covered this [in this post](#). It should also be noted that the [feedback node](#) in LabVIEW may be used as an equivalent to the shift register, although that may be a topic for another day!

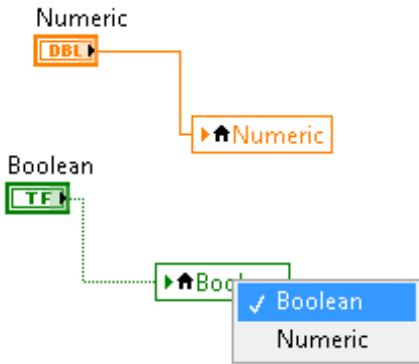
Introduction to Local and Global Variables

Local variables may be used within the same VI. Create a local variable by right-clicking a control or indicator on your Front Panel:

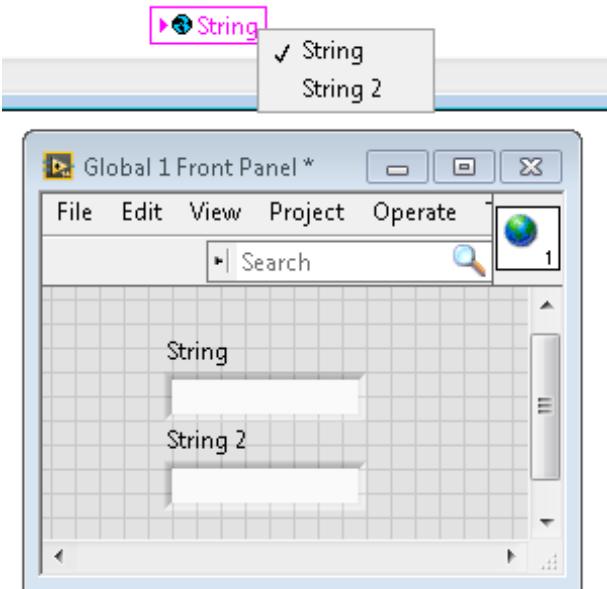


You may create a local variable from the Structures palette on the block diagram as well. When you have multiple local variables in one VI, you can left-click to choose which variable it is:

FRC LabVIEW Programming



Global variables are created slightly differently. Add one to the block diagram from the Structures palette, and notice that when you double-click it, it opens a separate front panel. This front panel does not have a block diagram, but you add as many entities to the front panel as you wish and save it as a *.vi file:

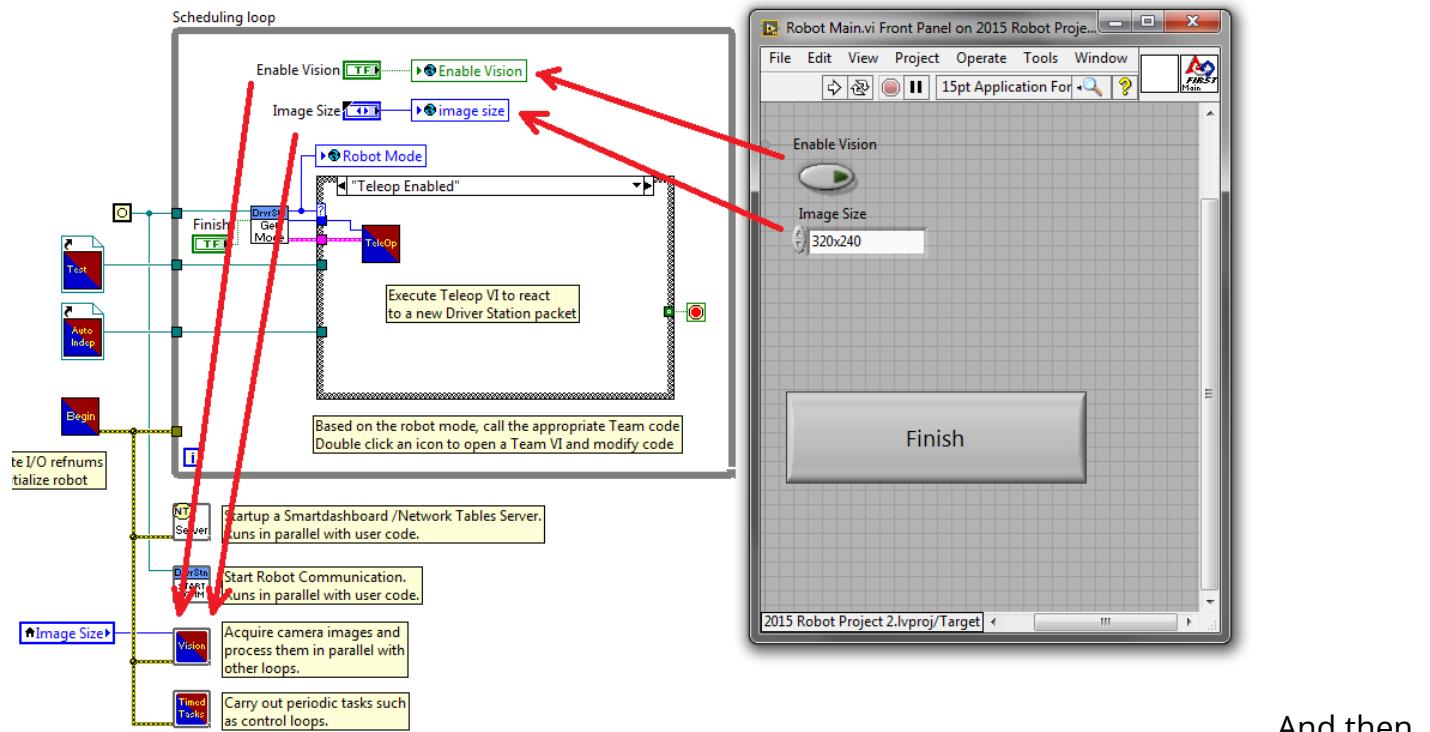


Note: Be very careful to avoid race conditions when using local and global variables! Essentially, make sure that you are not accidentally writing to the same variable in multiple locations without a way to know to which location it was last written. For a more thorough explanation, see this [help document](#).

How They are Used in the Default LabVIEW for FRC Robot Project

Global variables for “Enable Vision” and “Image Size” are written to during each iteration of the Robot Main VI...

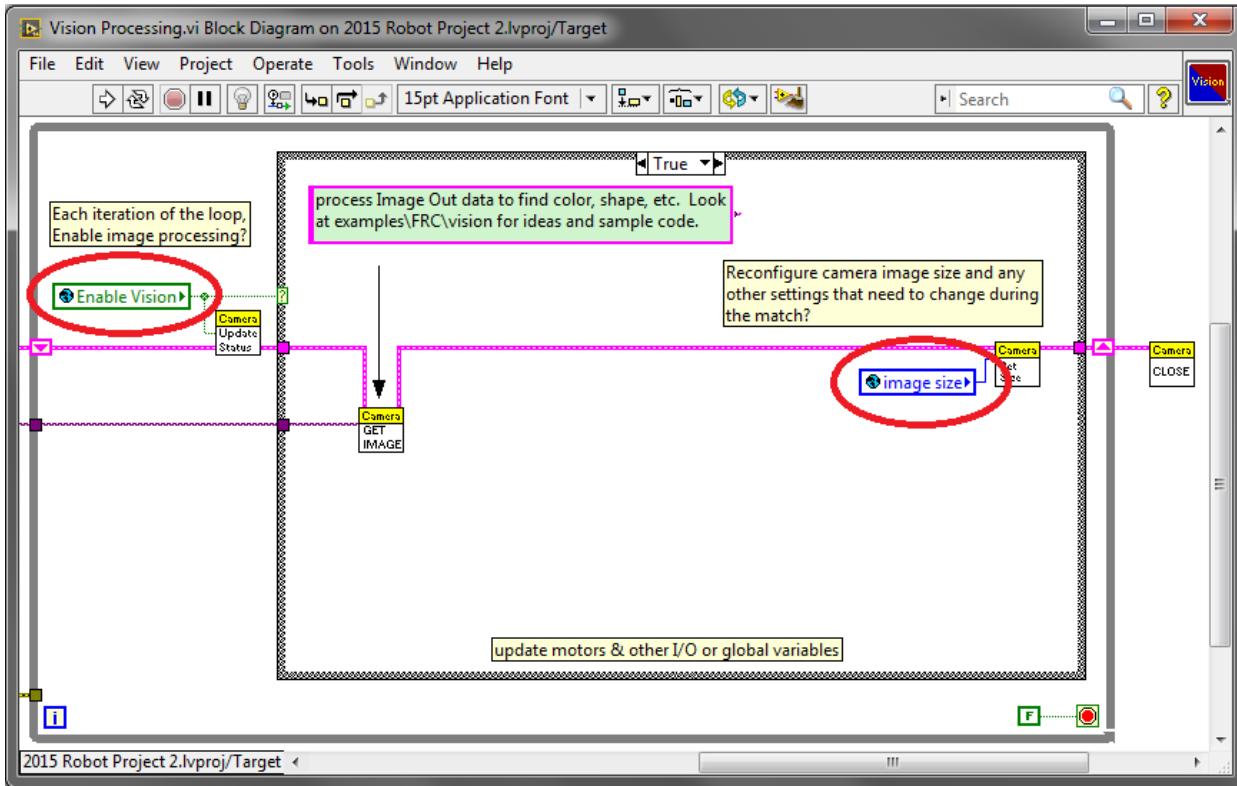
FRC LabVIEW Programming



...And then

read in each iteration of the Vision Processing VI:

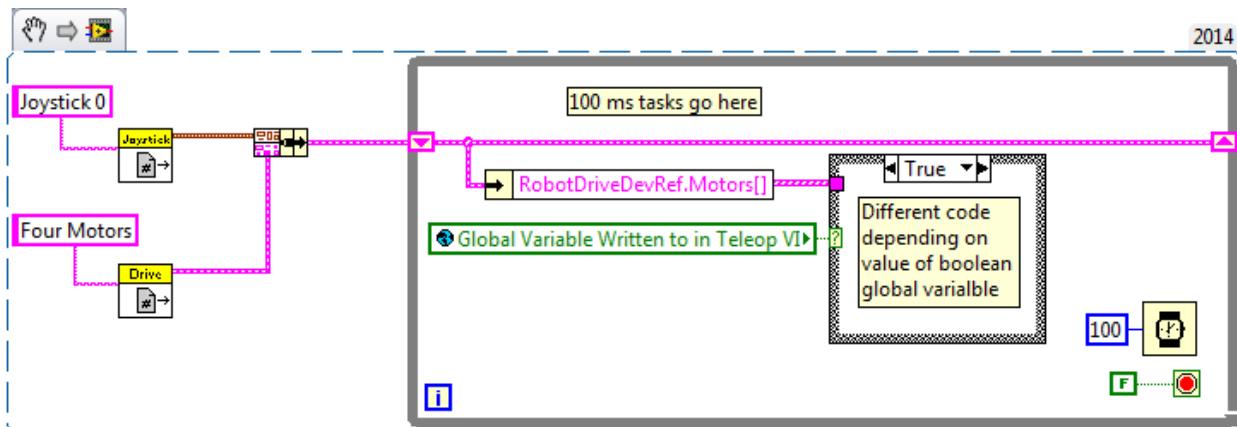
FRC LabVIEW Programming



This allows the user, when deploying to Robot Main VI from the LabVIEW Development Environment, to enable/disable vision and change the image size from Robot Main's Front Panel.

How Can You Use Them in Your Project?

Check out the block diagram for the Periodic Tasks VI. Perhaps there is some value, such as a boolean, that may be written to a global variable in the Teleop VI, and then read from in the Periodic Tasks VI. You can then decide what code or values to use in the Periodic Tasks VI, depending on the boolean global variable:



Using the Compressor in LabVIEW



This snippet shows how to set up your roboRIO project to use the Pneumatic Control Module (PCM). The PCM automatically starts and stops the compressor when specific pressures are measured in the tank. In your roboRIO program, you will need to add the following VIs.

For more information, check out the following links:

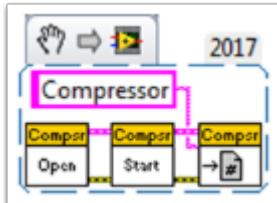
<https://rps01.usfirst.org/frc/manual/2015/2015FRCPneumaticsManual.pdf>

<http://crossstheroadelectronics.com/PCM%20User's%20Guide.pdf>

<http://team358.org/files/pneumatic/Pneumatics-StepByStep-roboRIO.pdf>

Begin VI

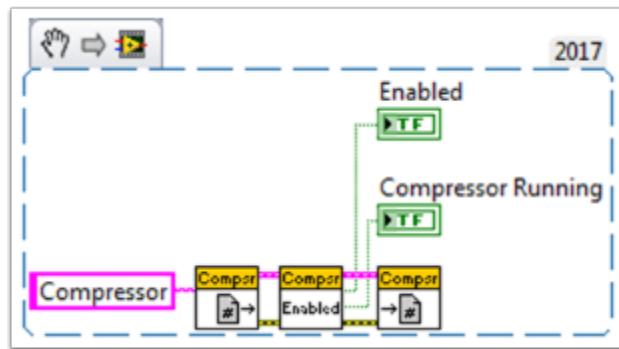
Place this snippet in the Begin.vi.



Teleop VI

Place this snippet in the Teleop.vi. This portion is only required if you are using the outputs for other processes.

FRC LabVIEW Programming



Finish VI

Place this snippet in Close Refs, save data, etc. frame of the Finish.vi.

