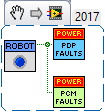
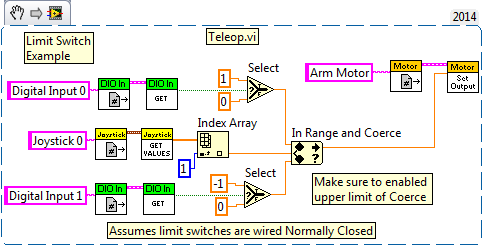
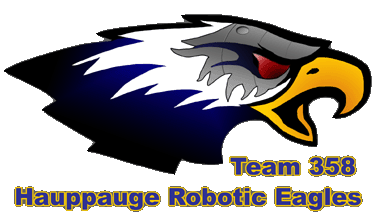
* **Clear PCM/PDP Sticky Faults** (2018) - This example uses the roboRIO User button to clear PCM and PDP sticky faults (changes the PDP/PCM status lights from yellow to green-until the next sticky fault occurs).   
  A good place for this is in Periodic Tasks in the 100ms loop.   
  WPI Robotics Library->Power   
  WPI Robotics Library->Utilities   
  
* **Limit Switch** (2015)   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->IO->DigitalInput   
  Programming->Structures   
  Programming->Comparison   
  Programming->Boolean   
  

[](http://www.team358.org/images/layout/logo/new-logo-large.png)

[](http://www.usfirst.org/)

**Upcoming Events:**

**Feb 19, 2018 7:00 AM:**  
Mid-winter Recess  
**Feb 20, 2018 11:59 PM:**  
Stop Build  
**Mar 1, 2018 8:00 AM:**  
Central NY Regional  
[Full event list](http://www.team358.org/events.php?date=1518370151)

***Team 358***

[Home](http://www.team358.org/)  
[Blog](http://www.team358.org/history/2018/)  
[Sponsors](http://www.team358.org/history/2018/sponsors.php)  
[News Articles](http://www.team358.org/history/articles/)  
[History](http://www.team358.org/history/)  
[Contact](http://www.team358.org/contact.php)

***FIRST*® *Things First***

[What is *FIRST*?](http://www.team358.org/first.php)  
[*FIRST* Official Site](http://www.firstinspires.org/)  
[*FIRST* in the News](http://www.firstinspires.org/about/first-in-the-news)  
[FRC® Game Details](http://www.team358.org/history/index.php#GameDetails)  
[FRC® Game Animations](http://team358.org/history/videos/GameAnimations.php)

***Multimedia***

[*FIRST* Promo Videos](http://team358.org/history/videos/)  
[*FIRST* News Segments](http://team358.org/history/videos/inthenews)  
[Team 358 Photos & Videos](http://www.team358.org/history/TeamMultimedia.php)  
[T-shirt Collection](http://www.team358.org/history/Tshirts/images.php?directory=Tshirts)

***Team Resources***

[Start a Team](http://www.team358.org/files/rookie)  
[Organization](http://www.team358.org/files/team_org)  
[Sponsorship](http://www.team358.org/files/sponsorship)  
[Fundraising](http://www.team358.org/files/fundraising)  
[Marketing/Publicity](http://www.team358.org/files/publicity)  
[Website](http://www.team358.org/files/website)  
[Strategy](http://www.team358.org/files/strategy)  
[Scouting](http://www.team358.org/files/scouting)  
[Career Planning](http://www.team358.org/files/career)  
[*FIRST* Scholarships](https://my.usfirst.org/scholarships/index.lasso)

***Technical Resources***

[Control System](http://www.team358.org/files/programming)  
[Mechanical](http://www.team358.org/files/mechanical)  
[Pneumatic](http://www.team358.org/files/pneumatic)  
[Electrical](http://www.team358.org/files/electrical)  
[Safety](http://www.team358.org/files/safety)  
[Workshops](http://www.team358.org/files/workshops)

***General Resources***

[Offsite Resources](http://www.team358.org/files/)  
[FRC® Statistics](http://www.team358.org/files/frc_records/index.php)  
[Long Island Teams](http://www.team358.org/files/li_teams/)

***FIRST*® Robotics Competition (FRC®)**

[**Control System 2015-2019**](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/)

**LabVIEW Programming**

[Game Data](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#GameData) | [Joystick](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Joystick) | [Digital Input](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#DigitalInput) | [Analog Input](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#AnalogInput) | [Relay](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Relay) | [Servo](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Servo) | [Single Motor](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#SingleMotor) | [Button Toggle](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#ButtonToggleAction) | | [Single Solenoid](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#SingleSolenoid) | [Double Solenoid](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#DoubleSolenoid) | [Arcade Drive](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#ArcadeDrive) | [Tank Drive](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#TankDrive) | [Mecanum Drive](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php" \l "MecanumDrive" \o ") | [CAN Drive](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#CANDrive) | [Nidec Brushless](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php" \l "NidecMotor" \o ") | [Button Start of Timed Action](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#ButtonStartofTimedAction) | [Which Button Was Pushed](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#WhichButtonWasPushed) | [Button Increment Power](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#ButtonIncrementPower) | [POV-Hat Switch](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#POV) | [Clear PCM/PDP Sticky Faults](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#StickyFaults) | [Switch](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Switch) | [Limit Switch](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#LimitSwitch) | [Limit Throttle](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#LimitThrottle) | [Limit Max Power](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#JoystickMaxThrottle) | [Joystick Response Curve](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#JoystickResponseCurve) | [Joystick Deadband](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#JoystickDeadband) | [Voltage Corrected Drive](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#VoltageCorrectedDrive) | [3-position Switch](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#3-positionSwitch) |[Digital BCD Switch](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#DigitalBCDSwitch) | [Write Data to Disk](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#WriteDatatoDisk) | [Write Image to Disk](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#WriteImagetoDisk) | [Tachometer](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Tachometer) | [Encoder](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Encoder) | [Gyroscope](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Gyroscope) | [Accelerometer](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Accelerometer) | [Potentiometer](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Potentiometer)| [Ultrasonic Sensor](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#UltrasonicSensor) | [I2CSensor](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#I2CSensor) | [Swap Robot Front](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#ReverseFront) | [Counting Things](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#CountingThings) | [Moving Average](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#MovingAverage) | [State Machine](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#StateMachine) | [Delayed Event](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#DelayedEvent) |[PID](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#PID) | [Camera](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Camera) | [Interrupt](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Interrupt) | [Equations](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Equations) | [Autonomous](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/index.php#Autonomous) |   
  
Training videos:   
  
LabVIEW Introduction-   
   
  
LabVIEW FUNdamentals, Episode 1 & 2-   
    
  
   
  
  
Resources:

* [LabVIEW Keyboard Shortcuts](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/LV_Quick_Reference.pdf)
* [WPI workshops](http://first.wpi.edu/FRC/contro876.html)
* [OpenG Libraries](http://sine.ni.com/nips/cds/view/p/lang/en/nid/209027)- Hundreds of Free, Reusable VIs
* [LabVIEW Wiki](http://labviewwiki.org/)
* [FRC\_Getting\_Started.pdf](http://team358.org/files/programming/ControlSystem2009-/LabVIEW/LabVIEW%20docs/FRC_Getting_Started.pdf)
* [FRC\_Programming\_Guide.pdf](http://team358.org/files/programming/ControlSystem2009-/LabVIEW/LabVIEW%20docs/FRC_Programming_Guide.pdf)
* [FRC\_Vision\_API\_Specification.pdf](http://team358.org/files/programming/ControlSystem2009-/LabVIEW/LabVIEW%20docs/FRC_Vision_API_Specification.pdf)
* [LabVIEW Development Environment system requirements](http://www.ni.com/labview/requirements)

**Special Topic Guides**

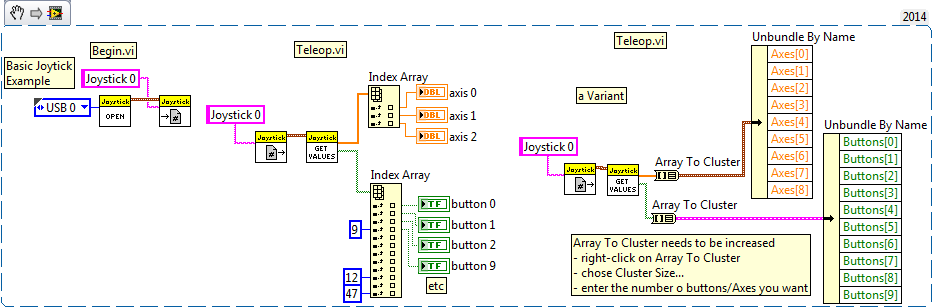
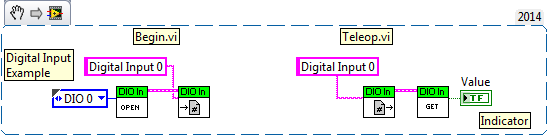
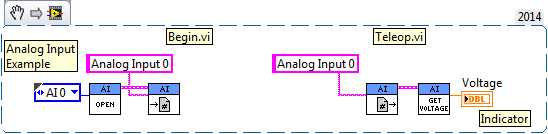
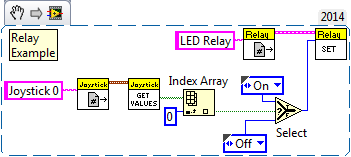
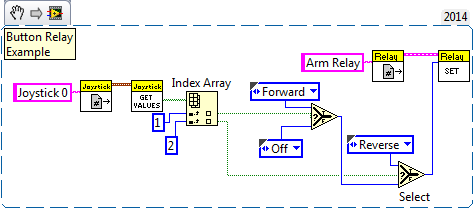
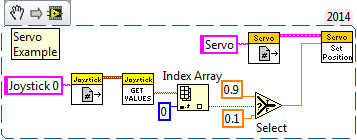
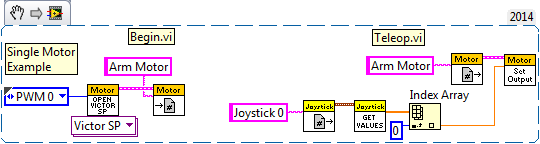
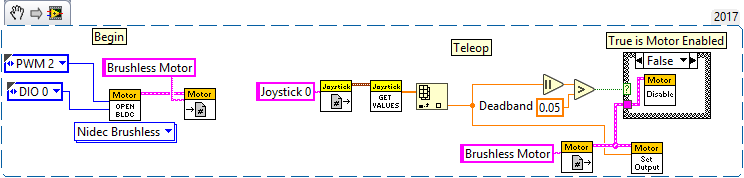
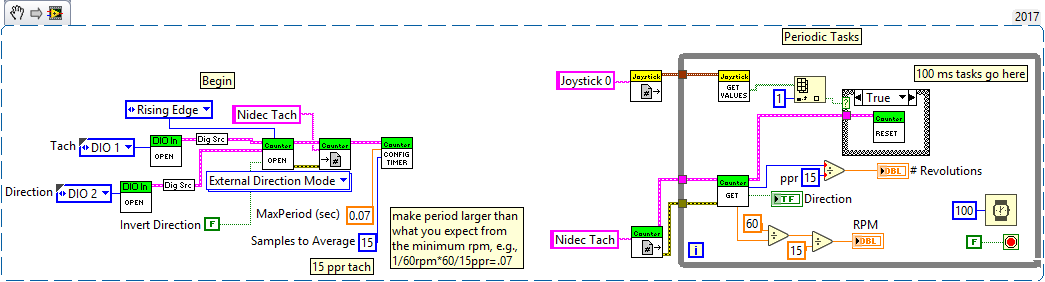
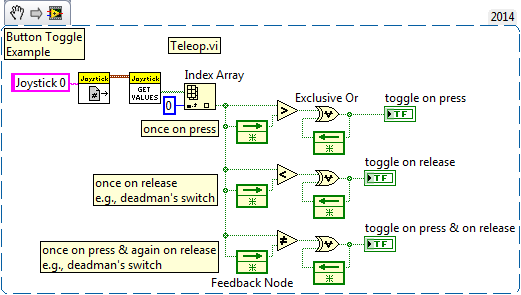
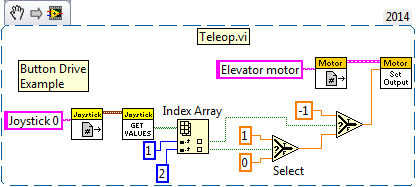
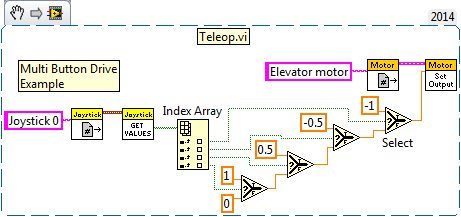
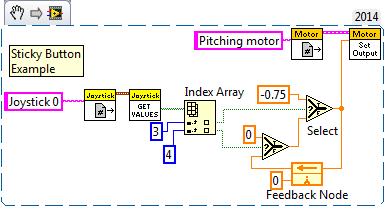
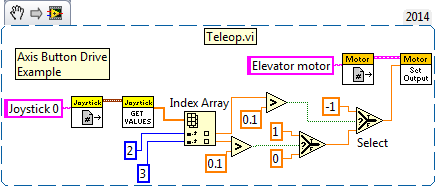
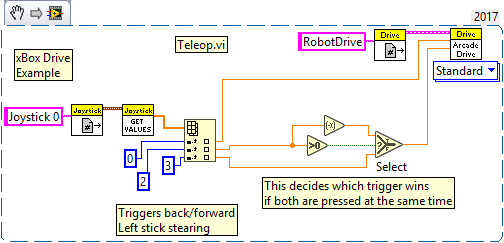
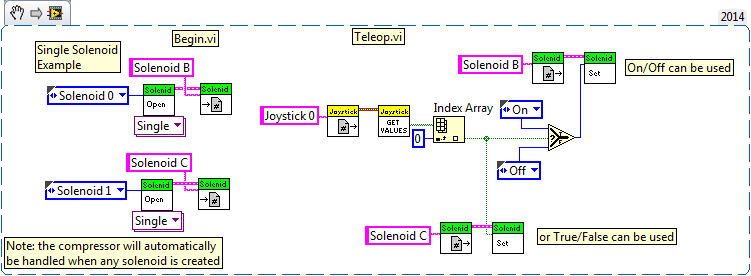
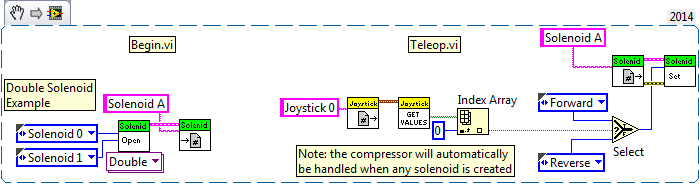
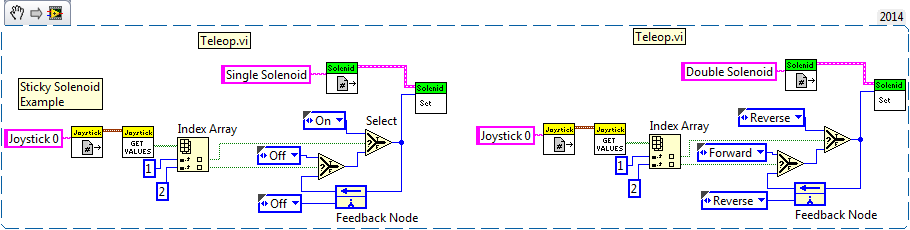
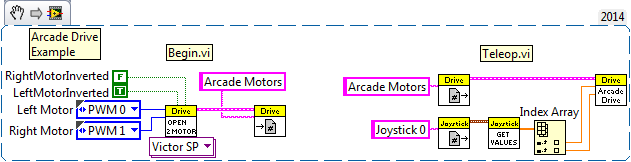
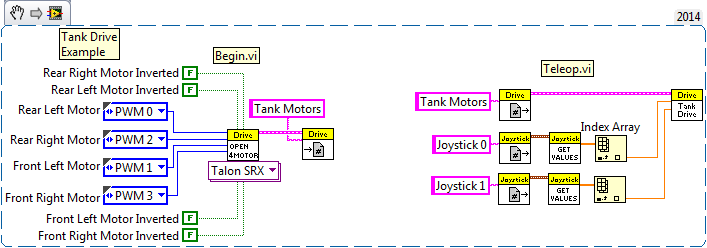
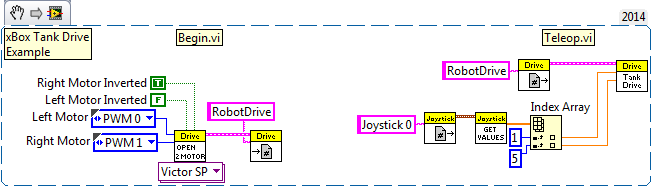
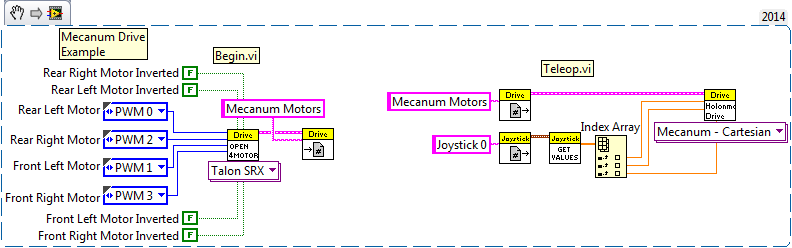
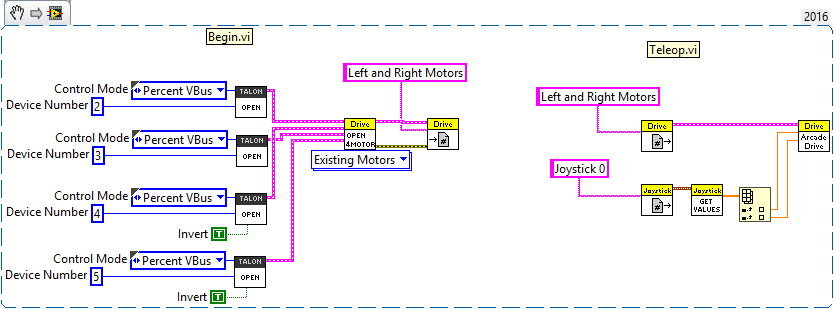
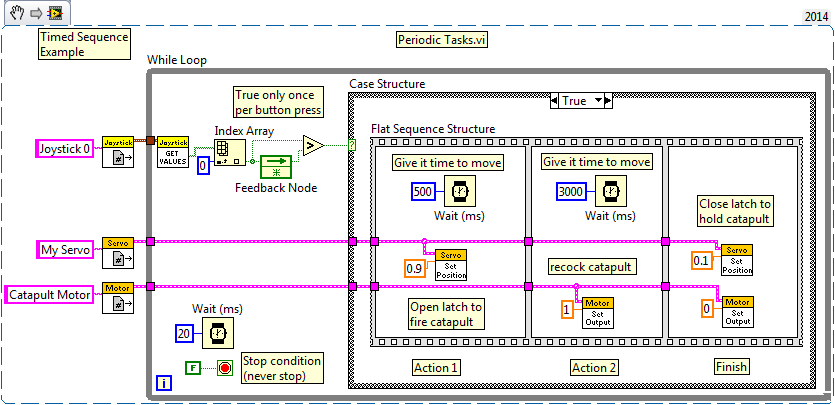
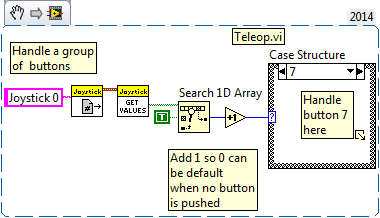
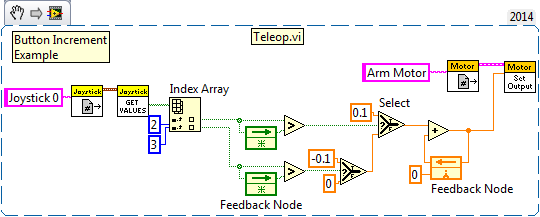
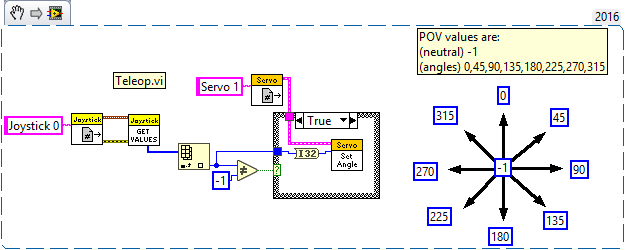
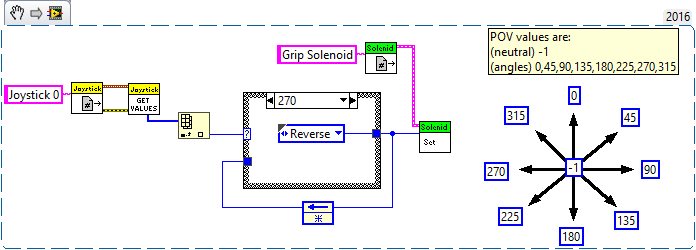
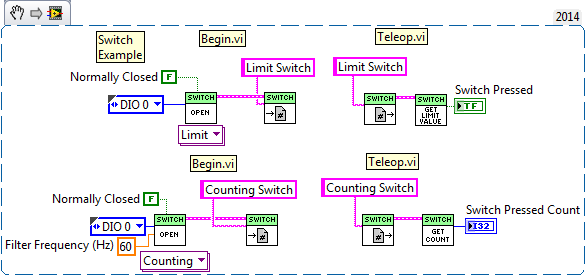
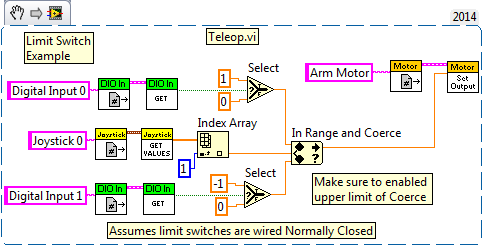
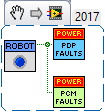
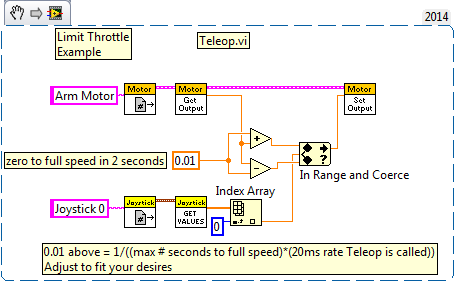
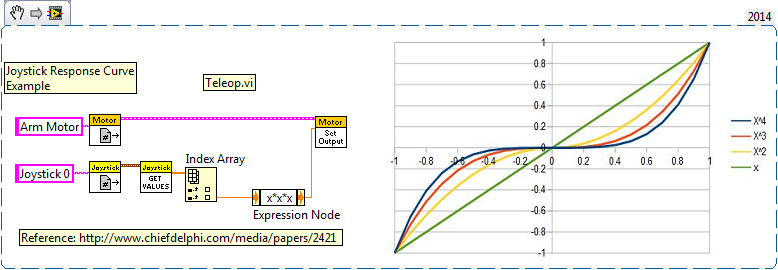
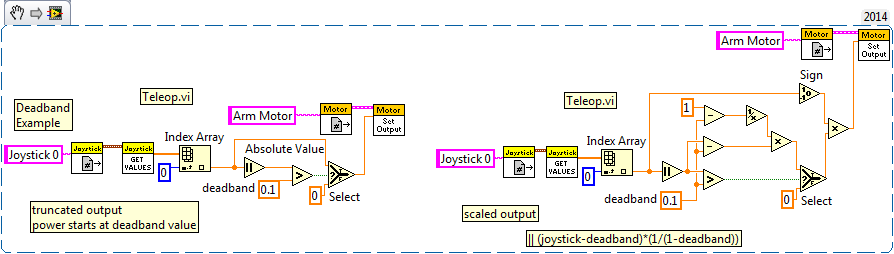
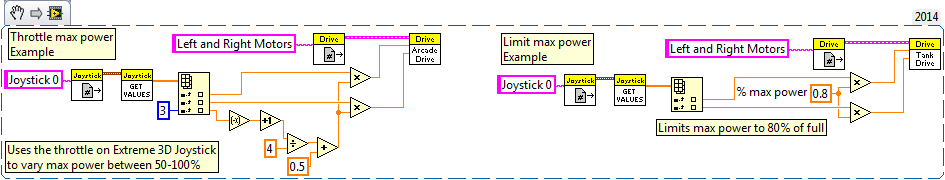
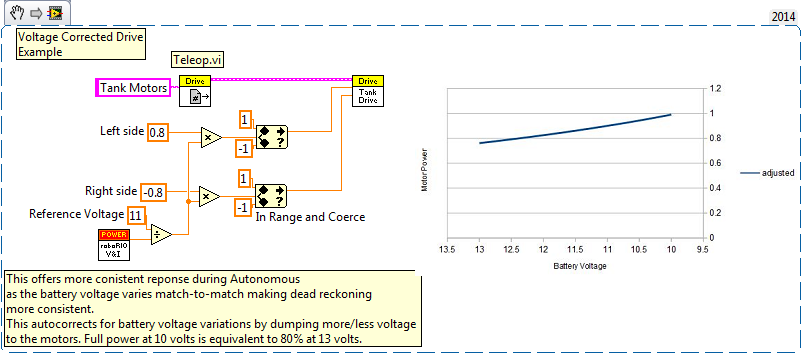
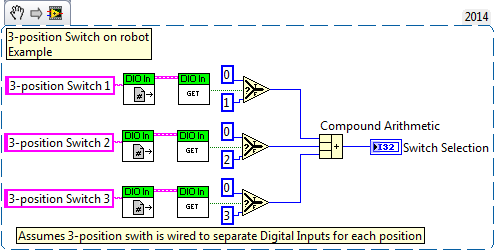
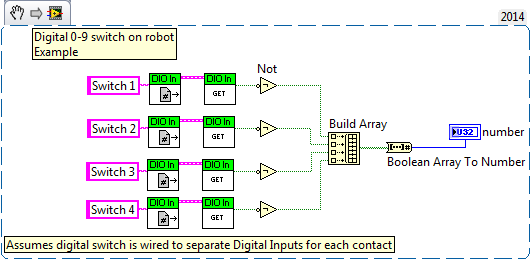
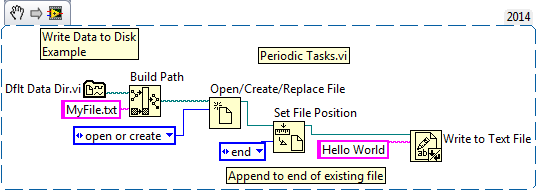
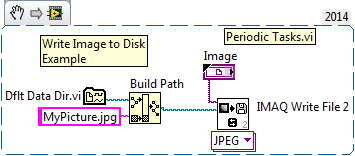
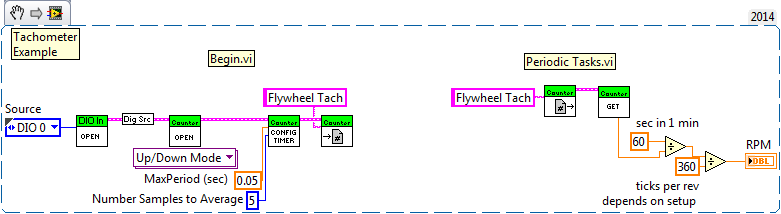
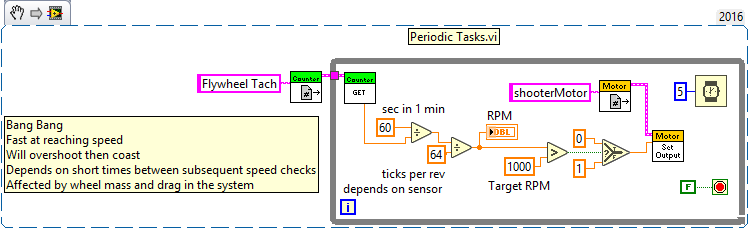
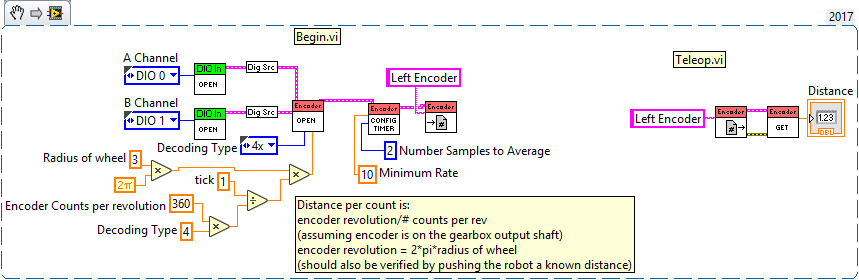
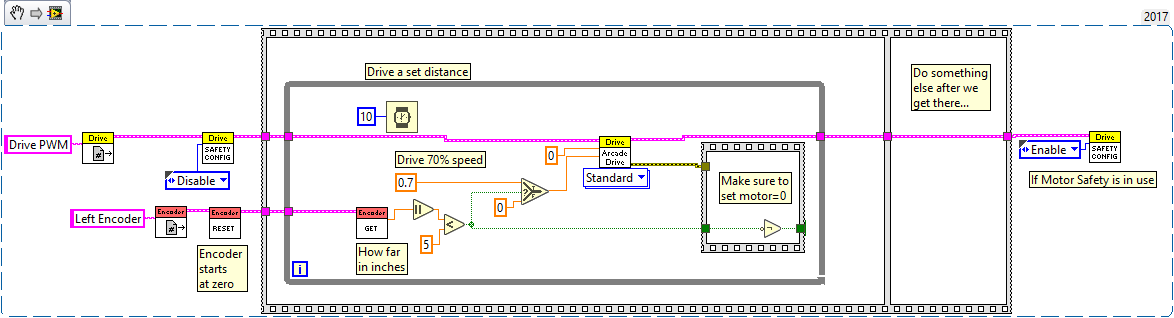
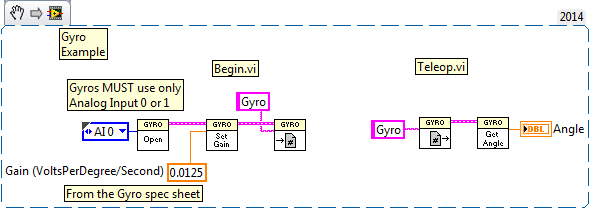
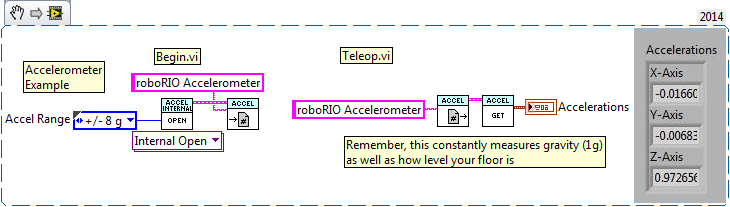
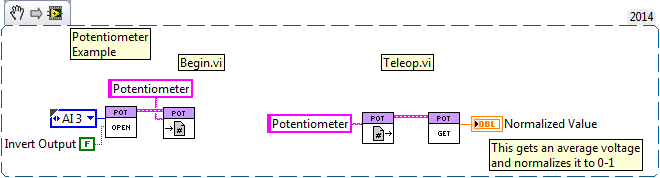
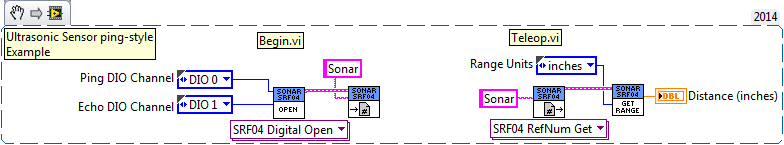
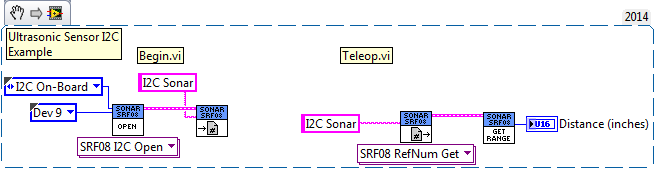
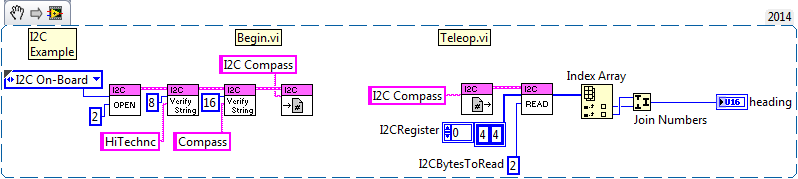
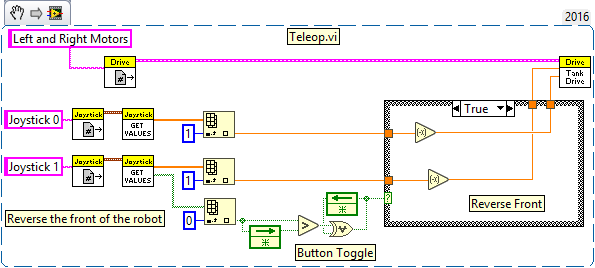
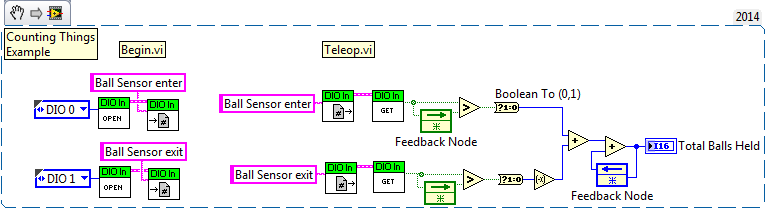
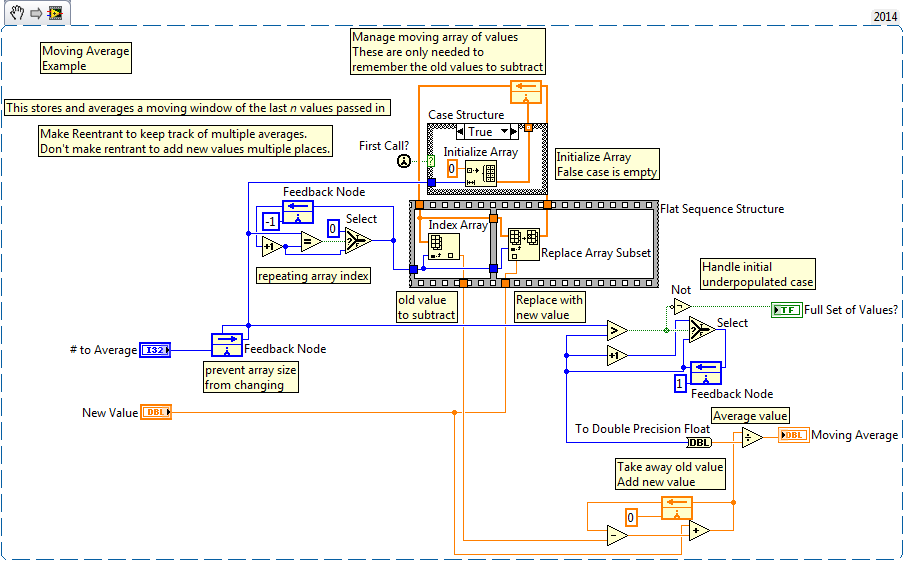
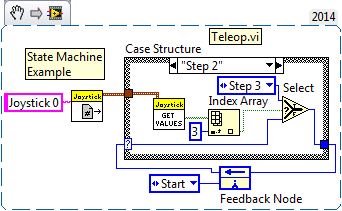
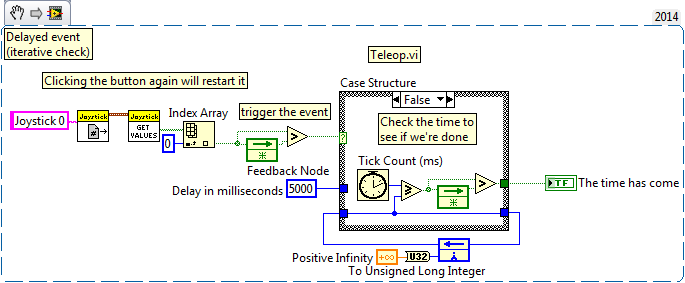
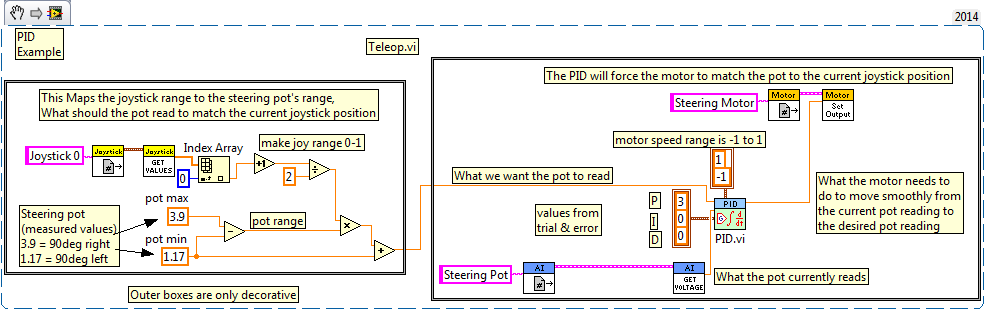
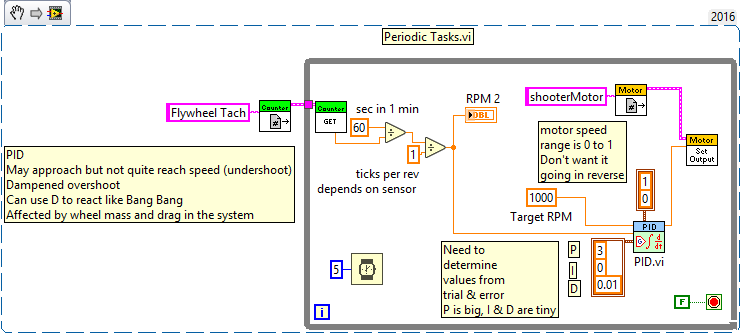
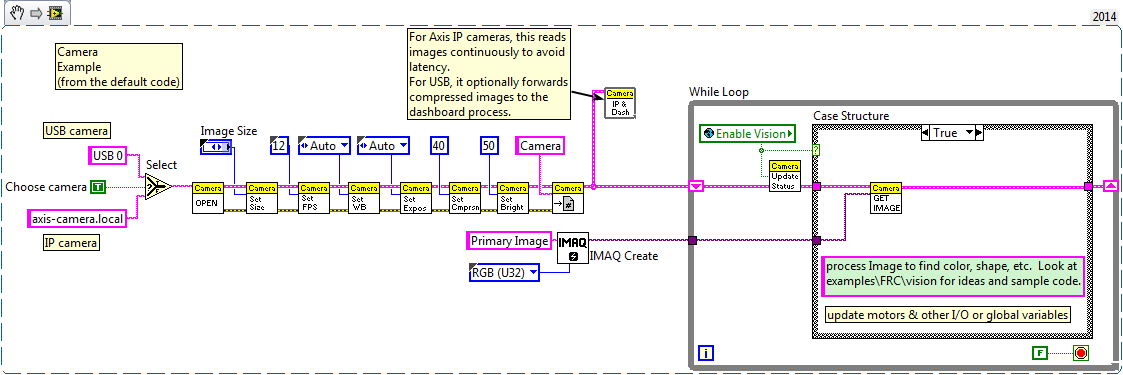
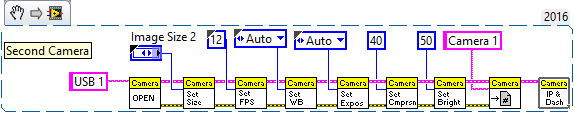
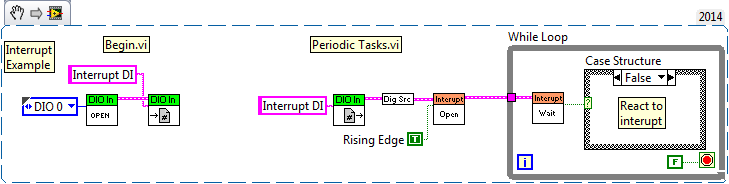
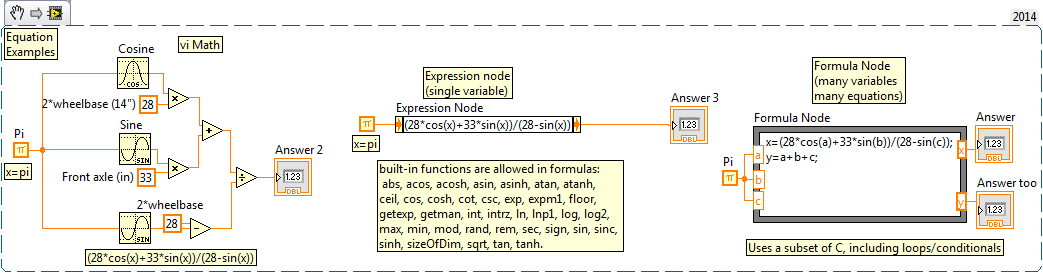
* [How to Create a vi](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/SubviExample.pdf)
* [Adding Global Variables](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/CreateGlobal.pdf)
* [Timing is Everything](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/Timing%20is%20Everything.pdf)
* [FRC LabVIEW Training Outline](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/LabVIEW_Training.doc) (rough initial draft)
* [Create Your Own FRC Dashboard](http://www.team358.org/files/programming/ControlSystem2015-2019/labview/HowtoCreateYourOwnDashboard.pdf)

[**LabVIEW Troubleshooting**](http://team358.org/files/programming/ControlSystem2009-/troubleshooting/indexLabVIEW.php)

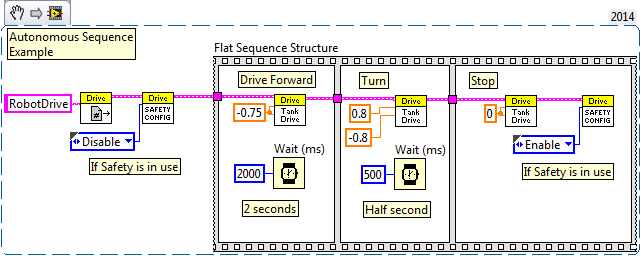
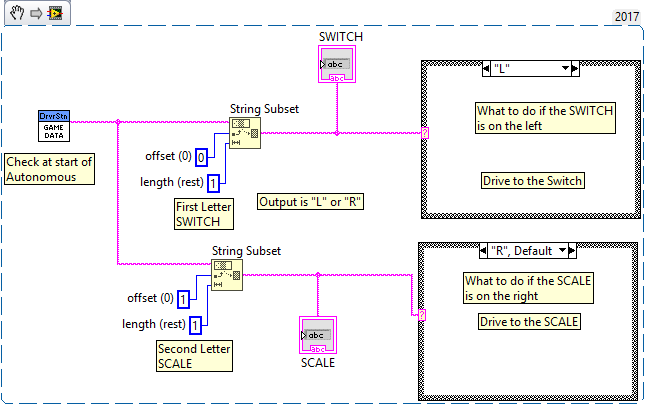
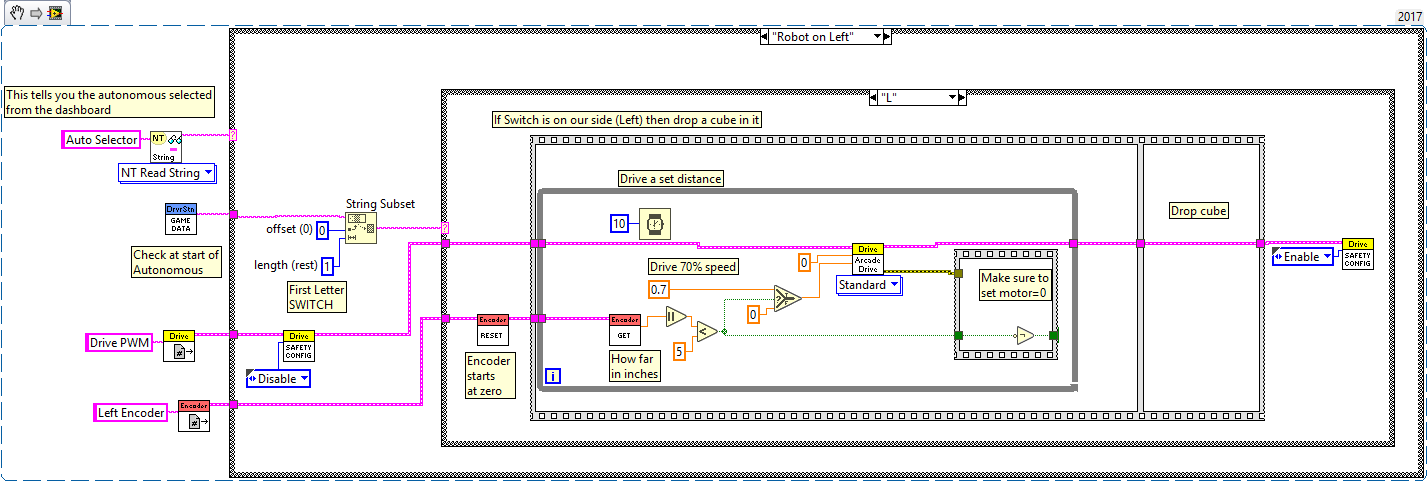
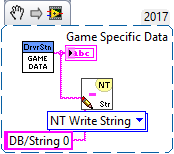
Special note on the library vi's: Starting in 2013 the cRIO simulator was added. This introduced two different versions of many of the WPILib vi's, one for executing stand-alone on a PC and another for standard operation on the robot cRIO. The proper version of the library code gets loaded depending on the current Target (PC vs. cRIO). An odd side effect of this is that if you drop a library vi onto a blank vi just to drill down to study the internals, then you will probably get the PC version and you'll be studying how the cRIO simulator works, not how the robot version of the code works. 

**Common Robot Operations**

Here are examples of ways to do some of the most common robot functions. Because the framework spreads the required elements over several files these examples are in image form to get all the parts together where you can see everything in one glance. Each example includes the menu paths to where all the icons used within that image are found.   
LabVIEW also provides stand-alone ready-to-execute examples from the Getting Started Window that include wiring diagrams for connecting devices properly. Those sample projects are great for testing if a device is working and wired correctly. Since they are stand-alone programs it takes some intermediate understanding of the default framework to figure out how to properly integrate the concepts into your LabVIEW project.

* **JoystickExample** (2015)   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Digital Input Example**, e.g., limit switches (2015)   
  WPI Robotics Library->IO->DigitalInput   
  
* **Analog Input Example** (2015)   
  WPI Robotics Library->IO->AnalogChannel   
  
* **Relay Example** (2015)   
  WPI Robotics Library->Actuators->Relay   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Comparison   
  Programming->Array->Index Array   
     
  
* **Servo Example** (2015)   
  WPI Robotics Library->Actuators->Servo   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Comparison   
  Programming->Array->Index Array   
  
* **Single Motor Example** (2015)   
  WPI Robotics Library->Actuators->MotorControl   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Nidec Brushless Motor Example** (2018)   
  WPI Robotics Library->Actuators->MotorControl   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  If the Nidec DIO connection comes loose, then the motor will race uncontrollably while the robot is Enabled.   
  Here is an example of how to minimize that potential problem. The motor is Disabled when neutral (whenever it is not actively being commanded to move).   
  
* **Nidec Brushless Motor Tachometer Example** (2018)   
  WPI Robotics Library->Actuators->MotorControl   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Button Toggle Action** (2015)   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Structures   
  Programming->Comparison   
  Programming->Cluster,Class,&Variant   
  Programming->Array->Index Array   
  
* **Button Control of Motor Example** (2015)   
  WPI Robotics Library->Actuators->MotorControl   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Comparison   
  Programming->Array->Index Array   
     
  Want more buttons? Add a Select for each like so:   
     
  Don't want to have to keep holding the button down? Then add a Feedback Node to remember:   
     
  Using an xBox where the triggers are actually read as an Axis:   
   
* **Single Solenoid Example** (2015)   
  WPI Robotics Library->Actuators->Solenoid   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Double Solenoid Example** (2015)   
  WPI Robotics Library->Actuators->Solenoid   
  Programming->Boolean   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
     
  
* **Arcade Drive Example** (2015)   
  WPI Robotics Library->RobotDrive   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Tank Drive Example** (2015)   
  WPI Robotics Library->RobotDrive   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
     
  
* **Holonomic or Mecanum Drive Example** (2015)   
  WPI Robotics Library->RobotDrive   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **CAN Arcade Drive Example** (2017)   
  WPI Robotics Library->Actuators->Motor Control->CAN Motor->Talon SRX   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Array->Index Array   
  
* **Button Start of Timed Action** (2015) - the timed-delays used here as an example can be replaced with sensors that tell positively when the mechanism is cocked or has finished shooting.   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Structures   
  Programming->Comparison   
  Programming->Array->Index Array   
  
* **Which Button Was Pushed** (2015) - Useful for deciding which of several joystick or custom Driver Station buttons was pushed.   
  This acts based on which of an array of buttons was pushed. Lowest number button wins if several are pushed at once.   
  WPI Robotics Library->DriverStation->Enhanced IO   
  Programming->Structures   
  Programming->Arrays   
  Programming->Numeric   
  Programming->Boolean   
  Programming->Array->Search 1D Array   
  
* **Button Increment Power** (2015) - This raises or lowers the power to a motor whenever one of the buttons is pushed, so it increments the power one step with each push.   
  Programming->Structures   
  Programming->Comparison   
  Programming->Array->Index Array   
  
* **POV (Point of View or Hat control)** (2017) - The POV or Hat switch is a common control switch on game controllers. It's handled in an unusual way, because it is a 360 degree button that sends the directional angle it is pressed (in 45 degree increments). Servos & solenoids are only used here by way of example.   
  Programming->Structures   
  Programming->Array->Index Array   
  WPI Library->Actuators->Servo   
  WPI Library->Actuators->Solenoid   
   
* **Switch** (2015) - has a toggle for selecting Normally-Closed or Normally-Open wiring. Has a counter mode that counts the number of times the switch is activated.   
  WPI Robotics Library->Sensors->Switch   
  Programming->Array->Index Array   
  
* **Limit Switch** (2015)   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->IO->DigitalInput   
  Programming->Structures   
  Programming->Comparison   
  Programming->Boolean   
  
* **Clear PCM/PDP Sticky Faults** (2018) - This example uses the roboRIO User button to clear PCM and PDP sticky faults (changes the PDP/PCM status lights from yellow to green-until the next sticky fault occurs).   
  A good place for this is in Periodic Tasks in the 100ms loop.   
  WPI Robotics Library->Power   
  WPI Robotics Library->Utilities   
  
* **Limit Throttle** (2015) - this example works ONLY in Teleop.vi because it depends on the regular 50Hz rate at which Teleop.vi gets called. To use it in Periodic Tasks, for instance, it would be called inside a loop and a loop delay would be added.   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->Actuators->MotorControl   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Joystick Response Curve** (2015) - example is a simple cubed response curve   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->Actuators->MotorControl   
  Programming->Comparison   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Joystick Deadband** (2015) - How to deal with a sloppy joystick that doesn't quite return to zero when released.   
  The first example has the same deadband, but truncates the motor output delivering between .1 and 1 (or -.1 and -1), so no .05 power ever shows up. This is good for devices, such as drivetrains, that won't move with less than .3 power anyway. It gives the joystick a little more range.   
  In the second example the joystick is ignored within the deadband (.1 to -.1 in this example) and outside that band delivers motor power from 0 to 1.   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->Actuators->MotorControl   
  Programming->Comparison   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Joystick Max Throttle Limit** (2015) - How to deal with a drive train that's too fast (well if you can't gear it down).   
  The first example uses the throttle on an Extreme 3D joystick as a variable control to limit max driving power to between 50-100%. A possible variation might be to allow more power to the Arcade rotate input.   
  In the second example the max power is limited to a hardcoded 80%.   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->Actuators->MotorControl   
  Programming->Comparison   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Voltage Corrected Tank Drive Example** (2015) - this is a simple way to make your drive controls response a little more consistent   
  WPI Robotics Library->RobotDrive   
  WPI Robotics Library->DriverStation->Joystick   
  
* **3-position Switch** (2015)   
  WPI Robotics Library->IO->DigitalInput   
  Programming->Comparison   
  Programming->Numeric   
  
* **Digital 0-9 BCD Switch** (2015)   
  WPI Robotics Library->IO->DigitalInput   
  Programming->Boolean   
  Programming->Array   
  
* **Write Data to Disk** (2015)   
  Programming->File I/O   
  Programming->File I/O->File Constants   
  
* **Write Image to Disk** (2015)   
  Programming->File I/O   
  Programming->File I/O->File Constants   
  FIRST Vision->Image Management   
  
* **Tachometer** (2015) - assumes a digital sensor, such as a retro-reflective light sensor, is used to detect a spinning object. One caution about Counter - it can produce scrambled or bogus values on startup, so be careful and examine the results when you first put it to use.   
  WPI Robotics Library->Sensors->Counter   
     
  
* **Encoder** (2015) Only allows for four encoders to be sampled at 4x, but many at 2X.   
  WPI Robotics Library->Sensors->Encoder   
  WPI Robotics Library->IO->DigitalInput   
     
  
* **Gyroscope** (2015)   
  WPI Robotics Library->Sensors->Gyro   
  
* **Accelerometer** (2015)   
  WPI Robotics Library->Sensors->Accelerometer   
  
* **Potentiometer** (2015)   
  WPI Robotics Library->Sensors-> Potentiometer   
  
* **Ultrasonic Sensor** - Only for paired emitter/sensor Vex-style (2015)   
  WPI Robotics Library->Sensors->Ultrasonic   
     
  
* **I2C Sensor** (2015)   
  WPI Robotics Library->Communications->I2C   
  Programming->Array   
  Programming->Numeric->Data Manipulation   
  Programming->Array->Index Array   
  
* **Reverse or Swap Which is the Front of Robot** (2017)   
  Programming->Structures   
  Programming->Boolean   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Counting Things** (2015)   
  WPI Robotics Library->IO->DigitalInput   
  Programming->Structures   
  Programming->Comparison   
  Programming->Boolean   
  
* **Moving Average** (2015) - You call this subvi to add a new value with each call and it produces the average over a window of the last *n* samples. You tell it the number of values (*n*) to average only with the first call, thereafter, that input is ignored. It also lets you know when you have reached a full set of values. This example illustrates initialization, sequencing to avoid clobbering values as you use them, safeguarding from changing inputs that cannot be changed, array manipulation, and creating a cycling index.   
  LabVIEW has quite a few built-in filters that do the work for you located in the Functions pallete under Signal Processing->Filters.   
  Programming->Array   
  Programming->Structures   
  Programming->Comparison   
  Programming->Numeric   
  Programming->Numeric->Conversion   
  Programming->Boolean   
  Programming->Synchronization   
  
* **State Machine** (2015) - the idea here is that you want to do different things based on a particular state your program is tracking. In this example you'd add some kind of activities for each state, such as activating solenoids or motors. States might change based on sensor feedback, such as a ball sensor.   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Structures   
  Programming->Comparison   
  Programming->Numeric   
  Programming->Array->Index Array   
  
* **Delayed Event** (2015) - this is for use within iterative vi's, such as Teleop, and simply might start an action, then stop it n seconds later.   
  This type of action can also be performed for you by the Elapsed Time vi.   
  WPI Robotics Library->DriverStation->Joystick   
  Programming->Structures   
  Programming->Comparison   
  Programming->Timing   
  Programming->Numeric   
  Programming->Numeric->Conversion   
  Programming->Array->Index Array   
  
* **P(ID) Example** (2015)- This code slaves a pot on the robot to the x-axis of a joystick via Proportional feedback. No I or D terms are used. Most of the work here is just the part that maps the full range of joystick movement to the steering motor potentiometer (pot) sensor. The PID part is pretty simple. The harder part is tuning the P coefficient by experimenting with the real robot.   
  WPI Robotics Library->DriverStation->Joystick   
  WPI Robotics Library->IO->AnalogChannel   
  WPI Robotics Library->Actuators->MotorControl   
  PID   
  Programming->Numeric   
  Programming->Array->Index Array   
     
  
* **Camera Example** (2015)   
  WPI Robotics Library->Camera   
  *FIRST* Vision->Image Management   
  Programming->Structures   
     
  
* **Interrupt** (2015) Use of this will be rare, but this can be used with either a Digital Input or an Analog Trigger.   
  WPI Robotics Library->Utilities->Interrupts   
  WPI Robotics Library->IO->DigitalInput   
  
* **Equations** (2015)   
  Programming->Numeric   
  Mathematics->Elementary & Special Functions->Trigonometric Functions   
  Programming->Structures   
     
  The formula node allows for the insertion of a pretty good subset of C. It not only lets you write equations, but loops and conditional code as well.
* **Calling C code** (2015)   
  To call C functions from LabVIEW you can create a C source library, then call into that library using a Call Library Function.   
  For simple code use the formula node mentioned just above. The formula node allows for the insertion of a pretty good subset of C. It not only lets you write equations, but loops and conditional code as well.

**Autonomous Programming**

Autonomous Independent.vi can be multiple parallel tasks. For example, the following two samples can co-exist quite comfortably in the same vi. Two independent sequences operating at the same time, one driving the robot while the other blinks a solenoid status light. This case is quite simple, however, it illustrates that much more complicated parallel independent tasking such as this is possible. Maybe a robot juggling while navigating a maze.   
  
**Flat Sequence** - Probably the simplest method of programming a fixed sequence of autonomous moves. Everything within a frame must complete before the next frame will start. It has the look of movie film. This example assumes any Enabled Motor Safety's have been Disabled:   
   
  
**Game Data** - this just parses the 2018 FMS Game Data so the code can make decisions based on the characters received. This type of code would be used in the Autonomous vi, but would require more logic to tell it the robot starting position and have decision logc that decides what to do.   
 >   
Here is a snippet that echos the GameData received back to the Default Dashboard on the Basic tab.   
This gets placed in the Periodic Tasks 100ms loop.   


[Get Adobe Acrobat Reader](http://www.adobe.com/products/acrobat/readstep2.html)   
[Get Microsoft Excel Viewer](http://www.microsoft.com/downloads/details.aspx?familyid=1cd6acf9-ce06-4e1c-8dcf-f33f669dbc3a&displaylang=en)   
[Get Microsoft PowerPoint Viewer](http://www.microsoft.com/downloads/details.aspx?familyid=048dc840-14e1-467d-8dca-19d2a8fd7485&displaylang=en)   
[Get WinZip](http://www.winzip.com/downwzeval.htm)

Copyright © 2005 - 2018 Hauppauge Robotics. All rights reserved.  
[About](http://www.team358.org/about.php) this site.

"); ?>