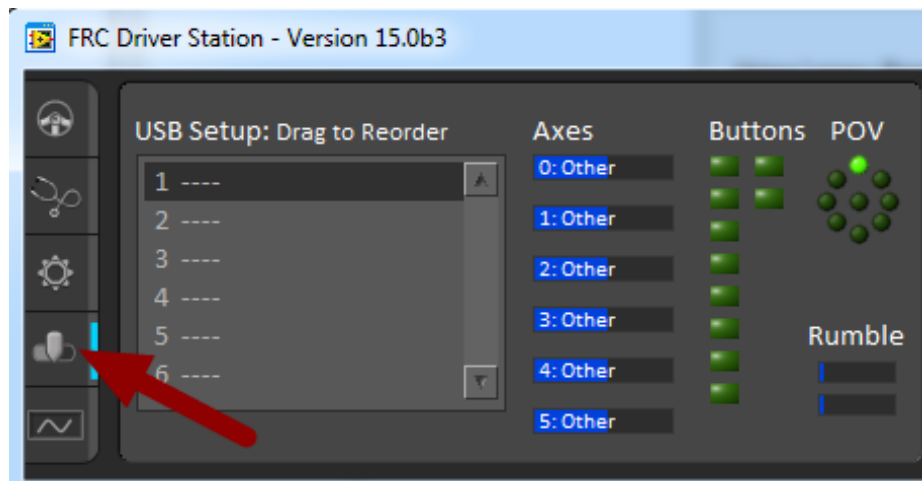


# Joysticks

## Joysticks

The standard input device supported by the WPI Robotics Library is a USB joystick or gamepad. The Logitech Attack 3 joystick provided in the KOP from 2009-2012 comes equipped with eleven digital input buttons and three analog axes, and interfaces with the robot through the Joystick class. The Joystick class itself supports joysticks with more capabilities as well such as the Logitech Extreme 3D Pro included in the 2013 KOP which has 4 analog axes and 12 buttons. Note that the rest of this article exclusively uses the term joystick but can also be referring to a HID compliant USB gamepad.

## USB connection



The joystick must be connected to one of the available USB ports on the driver station. The startup routine will read whatever position the joysticks are in as the center position, therefore, when the station is turned on the joysticks must be at their center position. In general the Driver Station software will try to preserve the ordering of devices between runs but it is a good idea to note what order your devices should be in and check each time you start the Driver Station software that they are correct. This can be done by selecting the USB Devices tab and viewing the order in the USB Setup box on the left hand side. Pressing a button on a joystick will cause its entry in the table to light up blue and have asterisks appear after the name. To reorder the joysticks simply click and drag.

# Joysticks

## Joystick Refresh

When the Driver Station is in disabled mode it is routinely looking for status changes on the joystick devices, unplugged devices are removed from the list and new devices are opened and added. When not connected to the FMS, unplugging a joystick will force the Driver Station into disabled mode. To start using the joystick again plug the joystick back in, check that it shows up in the right spot, then re-enable the robot. While the Driver Station is in enabled mode it will not scan for new devices as this is a time consuming operation and timely update of signals from attached devices takes priority.

When the robot is connected to the Field Management System at competition the Driver Station mode is dictated by the FMS. This means that you cannot disable your robot and the DS cannot disable itself in order to detect joystick changes. A manual complete refresh of the joysticks can be initiated by pressing the F1 key on the keyboard. Note that this will close and re-open all devices so all devices should be in their center position as noted above.

## Constructing a Joystick Object

### C++

```
Joystick *exampleStick

public:
    Robot() {
    }
    void RobotInit() {
        exampleStick = new Joystick(1);
    }
```

### Java

```
exampleStick = new Joystick(1);
```

# Joysticks

The primary constructor for the Joystick class takes a single parameter representing the port number of the Joystick, this is the number (1-6) next to the joystick in the Driver Station software's Joystick Setup box (shown in the first image). There is also a constructor which takes additional parameters of the number of axes and buttons and can be used with the get and set axis channel methods to create subclasses of Joystick to use with specific devices.

## Accessing Joystick Values - Option 1

### C++

```
double value;
value = exampleStick->GetX();
value = exampleStick->GetY();
value = exampleStick->GetZ();
value = exampleStick->GetThrottle();
value = exampleStick->GetTwist();

boolean buttonValue;
buttonValue = exampleStick->GetTop();
buttonValue = exampleStick->GetTrigger();
```

### Java

```
double value;
value = exampleStick.getX();
value = exampleStick.getY();
value = exampleStick.getZ();
value = exampleStick.getThrottle();
value = exampleStick.getTwist();

boolean buttonValue;
buttonValue = exampleStick.getTop();
buttonValue = exampleStick.getTrigger();
```

# Joysticks

There are two ways to access the current values of a joystick object. The first way is by using the set of named accessor methods or the `getAxis` method. The Joystick class contains the default mapping of these methods to the proper axes of the joystick for the KOP joystick. If you are using a another device you can subclass Joystick and use the `setAxisChannel` method to set the proper mappings if you wish to use these methods. Note that there are only named accessor methods for 5 of the 6 possible axes and 2 of the possible twelve buttons, if you need access to other axes or buttons, see Option 2 below.

Joystick axes return a scaled value in the range 1,-1 and buttons return a boolean value indicating their triggered state. Note that the typical convention for joysticks and gamepads is for Y to be negative as they joystick is pushed away from the user, "forward", and for X to be positive as the joystick is pushed to the right. To check this for a given device, see the section below on "Determining Joystick Mapping".

## Accessing Joystick Values - Option 2

### C++

```
double value;  
value = exampleStick->GetRawAxis(2);  
  
boolean buttonValue;  
buttonValue = exampleStick->GetRawButton(1);
```

### Java

```
double value;  
value = exampleStick.getRawAxis(1);  
  
boolean buttonValue;  
buttonValue = exampleStick.getRawButton(2);
```

# Joysticks

The second way to access joystick values is to use the methods `getRawAxis()` and `getRawButton()`. These methods take an integer representing the axis or button number as a parameter and return the corresponding value. For a method to determine the mapping between the physical axes and buttons of your device and the appropriate channel number see the section "Determining Joystick Mapping" below.

## Polar methods

### C++

```
double value;  
value = exampleStick->GetDirectionDegrees();  
value = exampleStick->GetDirectionRadians();  
value = exampleStick->GetMagnitude();
```

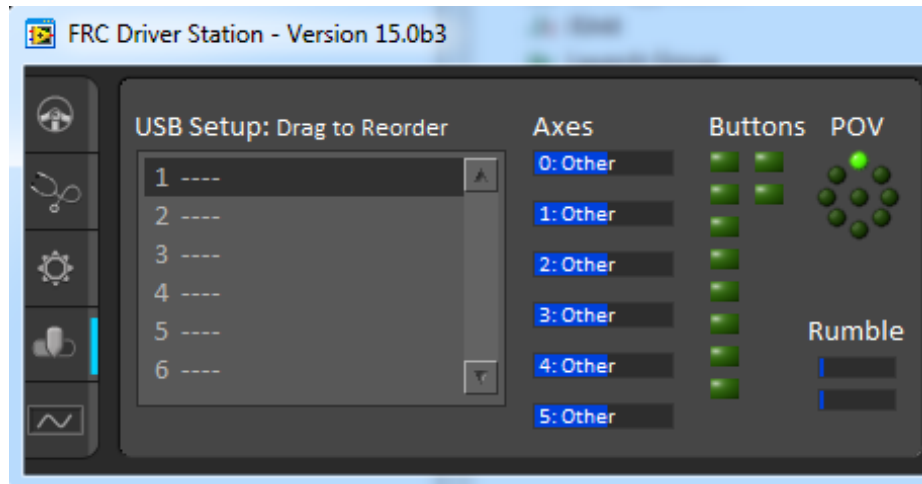
### Java

```
double value;  
value = exampleStick.getDirectionDegrees();  
value = exampleStick.getDirectionRadians();  
value = exampleStick.getMagnitude();
```

The Joystick class also contains helper methods for converting the joystick input to a polar coordinate system. For these methods to work properly, `getX` and `getY` have to return the proper axis (remap with `setChannel()` if necessary).

# Joysticks

## Determining Joystick Mapping



The 2015 FRC Driver Station contains indicators of the values of axes buttons and the POV that can be used to determine the mapping between physical joystick features and axis or button numbers. Simply click the joystick in the list to select it and the indicators will begin responding to the joystick input.