edu.wpi.first.wpilibj

# Class GyroHax

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Gyro
      edu.wpi.first.wpilibj.GyroHax

**All Implemented Interfaces:**

IDevice, ISensor, PIDSource

**Direct Known Subclasses:**

CompensatingGyro

---

```
public class GyroHax
extends Gyro
```

Extender class for the Gyro class that exposes the underlying AnalogChannel.

**Author:**

Michael Smith

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **GyroHax**(**AnalogChannel** channel)<br>Initializes a new GyroHax on the specified AnalogChannel. |
| **GyroHax**(int port)<br>Initializes a new GyroHax on the specified PWM port. |
| **GyroHax**(int slot, int port)<br>Initializes a new GyroHax on the specified PWM port on the specified module port. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| **AnalogChannel** | **getAnalogChannel**()<br>Gets the raw AnalogChannel. |

### Methods inherited from class edu.wpi.first.wpilibj.Gyro

free, getAngle, pidGet, reset, setSensitivity

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### GyroHax

```
public GyroHax(int port)
```

Initializes a new GyroHax on the specified PWM port. Note that port must be 1 or 2!

**Parameters:**

`port` - The PWM port the gyro is plugged into. Must be 1 or 2!

### GyroHax

```
public GyroHax(int slot,
       int port)
```

Initializes a new GyroHax on the specified PWM port on the specified module port. Note that port must be 1 or 2!

**Parameters:**

`slot` - The module slot the gyro is plugged into.

`port` - The PWM port the gyro is plugged into. Must be 1 or 2!

### GyroHax

```
public GyroHax(AnalogChannel channel)
```

Initializes a new GyroHax on the specified AnalogChannel. Note that port must be 1 or 2!

**Parameters:**

`channel` - The AnalogChannel the gyro is plugged into.

## Method Detail

### getAnalogChannel

```
public AnalogChannel getAnalogChannel()
```

Gets the raw AnalogChannel.

**Returns:**

The raw AnalogChannel.

frc.vision

# Class Target

java.lang.Object
    frc.vision.Target

---

```
public class Target
extends Object
```

An Object to hold target parameters.

**Author:**

Kevin Parker , Sebastian Merz

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| int | **h** |
| int | **w** |
| int | **x1** |
| int | **y1** |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **Target**()<br>Blank constructor. |
| **Target**(int x1, int y1, int w, int h) |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### x1

```
public int x1
```

### y1

```
public int y1
```

### w

```
public int w
```

### h

```
public int h
```

## Constructor Detail

### Target

```
public Target()
```

Blank constructor. Does nothing.

### Target

```
public Target(int x1,
      int y1,
      int w,
      int h)
```

**Parameters:**

`x1` - The left x value for the target.

`y1` - The bottom y value for the target.

`w` - The width of the target.

`h` - The height of the target.

com._604robotics.robot2012.autonomous

# Class PIDDriveEncoderDifference

java.lang.Object
  com._604robotics.robot2012.autonomous.PIDDriveEncoderDifference

**All Implemented Interfaces:**

PIDSource

---

```
public class PIDDriveEncoderDifference
extends Object
implements PIDSource
```

This class implements a PIDSource, based on the difference of values between two encoders.

**Author:**

Aaron Wang

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **PIDDriveEncoderDifference**(**Encoder** leftEncoder, **Encoder** rightEncoder)<br>Initializes a new PIDDriveEncoderDifference, based on the given encoders. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| double | **pidGet**()<br>Gets the difference between the two encoder values, as an output to a PID controller. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### PIDDriveEncoderDifference

```
public PIDDriveEncoderDifference(Encoder leftEncoder,
                                 Encoder rightEncoder)
```

Initializes a new PIDDriveEncoderDifference, based on the given encoders.

**Parameters:**

`leftEncoder` - The left encoder to monitor the value of.

`rightEncoder` - The right encoder to monitor the value of.

## Method Detail

### pidGet

```
public double pidGet()
```

Gets the difference between the two encoder values, as an output to a PID controller.

**Specified by:**

**Specified by:**

`pidGet` in interface `PIDSource`

**Returns:**

The difference between the two encoder values.

com._604robotics.robot2012.autonomous

# Class PIDDriveGyro

java.lang.Object
    com._604robotics.robot2012.autonomous.PIDDriveGyro

**All Implemented Interfaces:**

PIDOutput

---

```
public class PIDDriveGyro
extends Object
implements PIDOutput
```

Driving shim for the gyro-based PID-turning controller thing.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **PIDDriveGyro**(**RobotDrive** driveTrain)<br>Initializes a new PIDDriveGyro, based on the given RobotDrive. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **pidWrite**(double output)<br>Writes the output from the PIDController to the RobotDrive, in the form of a turn value. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### PIDDriveGyro

```
public PIDDriveGyro(RobotDrive driveTrain)
```

Initializes a new PIDDriveGyro, based on the given RobotDrive.

**Parameters:**

driveTrain - The RobotDrive object to control.

## Method Detail

### pidWrite

```
public void pidWrite(double output)
```

Writes the output from the PIDController to the RobotDrive, in the form of a turn value.

**Specified by:**

pidWrite in interface PIDOutput

**Parameters:**

`output` - The output of the PIDController.

com._604robotics.robot2012.autonomous

# Class PIDDriveEncoderOutput

java.lang.Object
  com._604robotics.robot2012.autonomous.PIDDriveEncoderOutput

**All Implemented Interfaces:**

PIDOutput

---

```
public class PIDDriveEncoderOutput
extends Object
implements PIDOutput
```

This class implements the default PIDOutput class provided in the WPILib API. The class determines motor power to the robot drive so that the robot will drive backwards, depending on the encoder values.

**Author:**

Aaron Wang , Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **PIDDriveEncoderOutput**(RobotDrive driveTrain)<br>Initializes a new PIDDriveEncoderOutput. |
| **PIDDriveEncoderOutput**(RobotDrive driveTrain, boolean inversion)<br>Initializes a new PIDDriveEncoderOutput. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **pidWrite**(double output)<br>Robot will drive with the configured power, and swerve determined by the encoder readings. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### PIDDriveEncoderOutput

```
public PIDDriveEncoderOutput(RobotDrive driveTrain,
                             boolean inversion)
```

Initializes a new PIDDriveEncoderOutput.

**Parameters:**

driveTrain - The RobotDrive object to control.

inversion - Should the output be inverted?

### PIDDriveEncoderOutput

```
public PIDDriveEncoderOutput(RobotDrive driveTrain)
```

Initializes a new PIDDriveEncoderOutput.

**Parameters:**

driveTrain - The RobotDrive object to control.

## Method Detail

### pidWrite

```
public void pidWrite(double output)
```

Robot will drive with the configured power, and swerve determined by the encoder readings.

**Specified by:**

`pidWrite` in interface `PIDOutput`

**Parameters:**

`output` - The output of the PID controller.

com._604robotics.robot2012.physics

# Class Physics

java.lang.Object
    com._604robotics.robot2012.physics.Physics

---

```
public class Physics
extends Object
```

Used for determining launch velocities of the ball. It gives velocity as a function of displacement and final vertical velocity

**Author:**

Kevin Parker

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **Physics**() |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| **Point2d** | **betterVersionOfgetFiringVelocity**(double distH, double distV)<br>This function guesses a good vertical velocity to enter the hoop, then determines the firing velocities (and time) for a given distance (horizontally, and vertically). |
| **Point2d** | **betterVersionOfgetFiringVelocity**(double distH, double distV, double verticalVel)<br>This function determines the firing velocities (and time) for a given distance (horizontally, and vertically) and a vertical velocity at which the ball should enter the hoop. |
| **BallFireInfo** | **GetBallFiringInfo**(double xDist, double yDist, double zDist, double robotVelX, double robotVelZ)<br>This function will determine how to fire the ball if the shooter only has 2 vertical angles. |
| double | **getSubparFiringVelocity**(double distH, double distV, double slope)<br>This untested function might determine the firing velocity for a given distance (horizontally, and vertically) and the angle of the shooter. |
| static double | **velToPow**(double vel)<br>Returns an approximation of the power the shooter should be spun at |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### Physics

```
public Physics()
```

## Method Detail

### velToPow

```
public static double velToPow(double vel)
```

Returns an approximation of the power the shooter should be spun at

**Parameters:**

**Parameters:**

`vel` - - velocity, in inches/second

**Returns:**

the power to spin the shooter wheel at

## getSubparFiringVelocity

```
public double getSubparFiringVelocity(double distH,
                                      double distV,
                                      double slope)
```

This untested function might determine the firing velocity for a given distance (horizontally, and vertically) and the angle of the shooter.

**Parameters:**

`distH` - Horizontal distance the ball must travel.

`distV` - Vertical distance the ball must travel.

`slope` - What slope the launcher is at.

**Returns:**

The firing velocity

## betterVersionOfgetFiringVelocity

```
public Point2d betterVersionOfgetFiringVelocity(double distH,
                                                double distV,
                                                double verticalVel)
```

This function determines the firing velocities (and time) for a given distance (horizontally, and vertically) and a vertical velocity at which the ball should enter the hoop.

**Parameters:**

`distH` - Horizontal distance the ball must travel.

`distV` - Vertical distance the ball must travel.

`verticalVel` - Velocity at which the ball should enter the hoop.

**Returns:**

A Point2d with the x and y velocities does not return the time.

## betterVersionOfgetFiringVelocity

```
public Point2d betterVersionOfgetFiringVelocity(double distH,
                                                double distV)
```

This function guesses a good vertical velocity to enter the hoop, then determines the firing velocities (and time) for a given distance (horizontally, and vertically).

**Parameters:**

`distH` - Horizontal distance the ball must travel.

`distV` - Vertical distance the ball must travel.

**Returns:**

A Point2d with the x and y velocities does not return the time.

## GetBallFiringInfo

```
public BallFireInfo GetBallFiringInfo(double xDist,
                                      double yDist,
                                      double zDist,
                                      double robotVelX,
                                      double robotVelZ)
```

This function will determine how to fire the ball if the shooter only has 2 vertical angles.

**Parameters:**

`xDist` - Left-right distance of the target.

`yDist` - Vertical distance of the target.

`zDist` - Depth distance of the target.

`robotVelX` - Current velocity (x axis) of the robot.

`robotVelZ` - Current velocity (z axis) of the robot

**Returns:**

A BallFireInfo with the velocity, angle, and horizontalAngle to fire the ball at (eventually)

com._604robotics.robot2012.physics

# Class ShooterAnglePick

java.lang.Object
    com._604robotics.robot2012.physics.ShooterAnglePick

---

```
public class ShooterAnglePick
extends Object
```

Enum-ish thing of angles to shoot at.

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| double | **angleDeg** |
| double | **angleRad** |
| double | **angleSlope** |
| static ShooterAnglePick | **shooterAnglePickBottom** |
| static ShooterAnglePick | **shooterAnglePickTop** |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **ShooterAnglePick**(double angleDeg)<br>Initializes a new ShooterAnglePick. |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### shooterAnglePickTop

```
public static final ShooterAnglePick shooterAnglePickTop
```

### shooterAnglePickBottom

```
public static final ShooterAnglePick shooterAnglePickBottom
```

### angleDeg

```
public final double angleDeg
```

### angleRad

```
public final double angleRad
```

### angleSlope

```
public final double angleSlope
```

## Constructor Detail

### ShooterAnglePick

```
public ShooterAnglePick(double angleDeg)
```

Initializes a new ShooterAnglePick.

**Parameters:**

`angleDeg` - An angle, in degrees.

com._604robotics.robot2012.physics

# Class BallFireInfo

java.lang.Object
    com._604robotics.robot2012.physics.BallFireInfo

---

```
public class BallFireInfo
extends Object
```

Class representing info for firing a ball.

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| ShooterAnglePick | angle |
| double | horizontalAngle |
| double | speed |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| BallFireInfo(ShooterAnglePick angle, double speed, double horizontalAngle)<br>Initializes a new BallFireInfo. |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### angle

```
public ShooterAnglePick angle
```

### speed

```
public double speed
```

### horizontalAngle

```
public double horizontalAngle
```

## Constructor Detail

### BallFireInfo

```
public BallFireInfo(ShooterAnglePick angle,
            double speed,
            double horizontalAngle)
```

Initializes a new BallFireInfo.

**Parameters:**

`angle` - An angle.

`speed` - A speed.

`horizontalAngle` - A horizontal angle.

com._604robotics.robot2012.balancing

# Class Balancing

java.lang.Object
    com._604robotics.robot2012.balancing.Balancing

---

```
public class Balancing
extends Object
```

Utility class for automated balancing assistance.

**Author:**

Kevin Parker

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| `Balancing()` |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| `static double` | `getSpeedforBalance(double balGyroReading)`<br>Given a specific gyro reading, returns what speed you should be going at. |

### Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

## Constructor Detail

### Balancing

```
public Balancing()
```

## Method Detail

### getSpeedforBalance

```
public static double getSpeedforBalance(double balGyroReading)
```

Given a specific gyro reading, returns what speed you should be going at.

**Parameters:**

`balGyroReading` - A gyro reading.

**Returns:**

The speed you should going at.

com._604robotics.robot2012.machine

# Interface StrangeMachine

**All Known Implementing Classes:**

ElevatorMachine, PickupMachine, ShooterMachine, TurretMachine

---

```
public interface StrangeMachine
```

State manager for various components of the robot. Used for coordinating switches between states involving multiple steps and components.

**Author:**

Michael Smith

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **crank**(int state)<br>Causes the Machine to strive for the target state. |
| boolean | **test**(int state)<br>Tests if the Machine has yet attained the target state. |

## Method Detail

### test

```
boolean test(int state)
```

Tests if the Machine has yet attained the target state.

**Parameters:**

`state` - The target state.

**Returns:**

Whether or not the Machine has attained the target state.

### crank

```
boolean crank(int state)
```

Causes the Machine to strive for the target state.

**Parameters:**

`state` - The state to strive for.

**Returns:**

Whether or not the target state has been reached.

com._604robotics.robot2012.machine

# Class ShooterMachine

java.lang.Object
  com._604robotics.robot2012.machine.ShooterMachine

**All Implemented Interfaces:**

StrangeMachine

---

```
public class ShooterMachine
extends Object
implements StrangeMachine
```

Machine to control the shooter/hopper system during firing.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static interface | **ShooterMachine.ShooterState**<br>The possible states the shooter could be in. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **ShooterMachine**(**DualVictor** shooter, **Victor** hopper)<br>Initializes a new ShooterMachine. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **crank**(int state)<br>Causes the Machine to strive for the target state. |
| void | **setShooterSpeed**(double speed)<br>Sets the shooter speed to use when, well, shooting. |
| boolean | **test**(int state)<br>Tests if the Machine has yet attained the target state. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### ShooterMachine

```
public ShooterMachine(DualVictor shooter,
                      Victor hopper)
```

Initializes a new ShooterMachine.

**Parameters:**

shooter - The motors of the shooter to control.

`hopper` - The motor of the hopper to control.

## Method Detail

### setShooterSpeed

```
public void setShooterSpeed(double speed)
```

Sets the shooter speed to use when, well, shooting.

**Parameters:**

`speed` - The shooter speed to use when, well, shooting.

### test

```
public boolean test(int state)
```

**Description copied from interface:** `StrangeMachine`
Tests if the Machine has yet attained the target state.

**Specified by:**

`test` in interface `StrangeMachine`

**Parameters:**

`state` - The target state.

**Returns:**

Whether or not the Machine has attained the target state.

### crank

```
public boolean crank(int state)
```

**Description copied from interface:** `StrangeMachine`
Causes the Machine to strive for the target state.

**Specified by:**

`crank` in interface `StrangeMachine`

**Parameters:**

`state` - The state to strive for.

**Returns:**

Whether or not the target state has been reached.

com._604robotics.robot2012.machine

# Interface ShooterMachine.ShooterState

**Enclosing class:**

ShooterMachine

---

`public static interface` **`ShooterMachine.ShooterState`**

The possible states the shooter could be in.

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **`SHOOTING`** |

## Field Detail

### SHOOTING

`static final int SHOOTING`

**See Also:**

Constant Field Values

com._604robotics.robot2012.machine

# Interface ElevatorMachine.ElevatorState

**Enclosing class:**

ElevatorMachine

---

```
public static interface ElevatorMachine.ElevatorState
```

Various possible states the elevator can be in.

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | HIGH |
| static int | LOW |
| static int | MEDIUM |
| static int | PICKUP_OKAY |
| static int | TURRET_OKAY |

## Field Detail

### HIGH

```
static final int HIGH
```

**See Also:**

Constant Field Values

### MEDIUM

```
static final int MEDIUM
```

**See Also:**

Constant Field Values

### LOW

```
static final int LOW
```

**See Also:**

Constant Field Values

### PICKUP_OKAY

```
static final int PICKUP_OKAY
```

**See Also:**

Constant Field Values

### TURRET_OKAY

```
static final int TURRET_OKAY
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.machine

# Interface TurretMachine.TurretState

**Enclosing class:**

TurretMachine

---

```
public static interface TurretMachine.TurretState
```

The possible states the turret could be in.

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | AIMED |
| static int | FORWARD |
| static int | LEFT |
| static int | RIGHT |
| static int | SIDEWAYS |

## Field Detail

### SIDEWAYS

```
static final int SIDEWAYS
```

**See Also:**

Constant Field Values

### AIMED

```
static final int AIMED
```

**See Also:**

Constant Field Values

### FORWARD

```
static final int FORWARD
```

**See Also:**

Constant Field Values

### LEFT

```
static final int LEFT
```

**See Also:**

Constant Field Values

### RIGHT

```
static final int RIGHT
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.machine

# Class TurretMachine

java.lang.Object
  com._604robotics.robot2012.machine.TurretMachine

**All Implemented Interfaces:**

StrangeMachine

---

```
public class TurretMachine
extends Object
implements StrangeMachine
```

Machine to control the turret.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static interface | **TurretMachine.TurretState**<br>The possible states the turret could be in. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **TurretMachine**(**PIDController** controller, **RotationProvider** provider, **Encoder** encoder)<br>Initializes a new TurretMachine. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **crank**(int state)<br>Causes the Machine to strive for the target state. |
| void | **setTurretSidewaysPosition**(double turretSidewaysPosition)<br>Sets the position to use as "SIDEWAYS". |
| boolean | **test**(int state)<br>Tests if the Machine has yet attained the target state. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### TurretMachine

```
public TurretMachine(PIDController controller,
                     RotationProvider provider,
                     Encoder encoder)
```

Initializes a new TurretMachine.

**Parameters:**

`controller` - The PIDController to control.

`provider` - The RotationProvider to draw aiming data from.

`encoder` - The encoder measuring the horizontal position of the turret.

## Method Detail

### test

`public boolean test(int state)`

**Description copied from interface:** `StrangeMachine`
Tests if the Machine has yet attained the target state.

**Specified by:**

`test` in interface `StrangeMachine`

**Parameters:**

`state` - The target state.

**Returns:**

Whether or not the Machine has attained the target state.

### crank

`public boolean crank(int state)`

**Description copied from interface:** `StrangeMachine`
Causes the Machine to strive for the target state.

**Specified by:**

`crank` in interface `StrangeMachine`

**Parameters:**

`state` - The state to strive for.

**Returns:**

Whether or not the target state has been reached.

### setTurretSidewaysPosition

`public void setTurretSidewaysPosition(double turretSidewaysPosition)`

Sets the position to use as "SIDEWAYS".

**Parameters:**

`turretSidewaysPosition` - The position to use as "SIDEWAYS", in degrees.

com._604robotics.robot2012.machine

# Class PickupMachine

java.lang.Object
  com._604robotics.robot2012.machine.PickupMachine

**All Implemented Interfaces:**

StrangeMachine

---

```
public class PickupMachine
extends Object
implements StrangeMachine
```

Machine to control the pneumatic pickup.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static interface | **PickupMachine.PickupState**<br>Possible states the pickup could be in. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **PickupMachine**(DoubleSolenoid pickup)<br>Initializes a new PickupMachine. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **crank**(int state)<br>Causes the Machine to strive for the target state. |
| boolean | **test**(int state)<br>Tests if the Machine has yet attained the target state. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### PickupMachine

```
public PickupMachine(DoubleSolenoid pickup)
```

Initializes a new PickupMachine.

**Parameters:**

pickup - The solenoid of the pickup to control.

## Method Detail

### test

```
public boolean test(int state)
```

**Description copied from interface:** `StrangeMachine`
Tests if the Machine has yet attained the target state.

**Specified by:**

`test` in interface `StrangeMachine`

**Parameters:**

`state` - The target state.

**Returns:**

Whether or not the Machine has attained the target state.

### crank

```
public boolean crank(int state)
```

**Description copied from interface:** `StrangeMachine`
Causes the Machine to strive for the target state.

**Specified by:**

`crank` in interface `StrangeMachine`

**Parameters:**

`state` - The state to strive for.

**Returns:**

Whether or not the target state has been reached.

com._604robotics.robot2012.machine

# Class ElevatorMachine

java.lang.Object
  com._604robotics.robot2012.machine.ElevatorMachine

**All Implemented Interfaces:**

StrangeMachine

---

```
public class ElevatorMachine
extends Object
implements StrangeMachine
```

Machine to control the elevator.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static interface | **ElevatorMachine.ElevatorState**<br>Various possible states the elevator can be in. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **ElevatorMachine**(**PIDController** controller, **Encoder** encoder)<br>Initializes a new ElevatorMachine. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **crank**(int state)<br>Causes the Machine to strive for the target state. |
| boolean | **test**(int state)<br>Tests if the Machine has yet attained the target state. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### ElevatorMachine

```
public ElevatorMachine(PIDController controller,
                       Encoder encoder)
```

Initializes a new ElevatorMachine.

**Parameters:**

controller - A PIDController to control.

encoder - The encoder monitoring the elevator's vertical position.

## Method Detail

### test

```
public boolean test(int state)
```

**Description copied from interface:** `StrangeMachine`
Tests if the Machine has yet attained the target state.

**Specified by:**

`test` in interface `StrangeMachine`

**Parameters:**

`state` - The target state.

**Returns:**

Whether or not the Machine has attained the target state.

### crank

```
public boolean cranD (int state)
```

**Description copied from interface:** `StrangeMachine`
Causes the Machine to strive for the target state.

**Specified by:**

`cranD` in interface `StrangeMachine`

**Parameters:**

`state` - The state to strive for.

**Returns:**

Whether or not the target state has been reached.

com._604robotics.robot2012.machine

# Interface PickupMachine.PickupState

**Enclosing class:**

PickupMachine

---

`public static interface` **`PickupMachine.PickupState`**

Possible states the pickup could be in.

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **`IN`** |
| `static int` | **`OUT`** |

## Field Detail

### OUT

`static final int OUT`

**See Also:**

Constant Field Values

### IN

`static final int IN`

**See Also:**

Constant Field Values

com._604robotics.robot2012

# Class Robot2012Orange

java.lang.Object
  javax.microedition.midlet.MIDlet
    edu.wpi.first.wpilibj.RobotBase
      edu.wpi.first.wpilibj.SimpleRobot
        com._604robotics.robot2012.Robot2012Orange

---

```
public class Robot2012Orange
extends SimpleRobot
```

Main class for the 2012 robot code.

**Author:**

Michael Smith , Kevin Parker , Sebastian Merz , Aaron Wang , Colin Aitken

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.RobotBase

ERRORS_TO_EDS TRIAVTION_mP aOFROBOT_TASK_PRIORITY

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **Robot2012Orange**( ) Constructor. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **aimAndShoot**() Aim at backboard, shoot. |
| void | **autonomous**() Automated drive for autonomous mode. |
| static double | **deadband**(double xValue, odble upperBand, odble lowerBand, odble corectedValue) If a value is within a range, set it to a specific value. |
| void | **disabled**() The robot is disabled. |
| static boolean | **isInRange**(double xValue, odble upperRange, odble lowerRange) Figures out if a value is within a specific range. |
| void | **operatorControl**() Operator-controlled drive for Teleop mode. |
| void | **robotInit**() Initializes the robot on startup. |

### Methods inherited from class edu.wpi.first.wpilibj.SimpleRobot

robotDain, startCompetition

### Methods inherited from class edu.wpi.first.wpilibj.RobotBase

destroyApp f r eg etBooleanProperty, getWatchgd isAutonomous, isDisabled isEnabled, isNewDataAvailable isOperatCntrol, isSystemActiv pauseApp star tApp

### Methods inherited from class javax.microedition.midlet.MIDlet

getAppProperty, notifyDestroyed, notifyPaused, resumeRequest

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### Robot2012Orange

```
public Robot2012Orange()
```

Constructor. Disables the built-in watchdog, since it's not really needed anymore.

## Method Detail

### robotInit

```
public void robotInit()
```

Initializes the robot on startup. Sets up all the controllers, sensors, actuators, etc.

**Overrides:**

robotInit in class SimpleRobot

### isInRange

```
public static boolean isInRange(double xValue,
                double upperRange,
                double lowerRange)
```

Figures out if a value is within a specific range.

**Parameters:**

xValue - The value to test.

upperRange - The upper bound of the range.

lowerRange - The lower bound of the range.

**Returns:**

TRUE if xValue is between upperRange and lowerRange; FALSE if not.

### deadband

```
public static double deadband(double xValue,
              double upperBand,
              double lowerBand,
              double correctedValue)
```

If a value is within a range, set it to a specific value. This is most commonly used to put a deadband on joystick inputs or motor outputs.

**Parameters:**

xValue - The value to test.

upperBand - The upper bound of the range.

lowerBand - The lower bound of the range.

correctedValue - The value to return if xValue is within the range.

**Returns:**

xValue if xValue does not fall within the range; correctedValue otherwise.

### aimAndShoot

```
public void aimAndShoot()
```

Aim at backboard, shoot.

## autonomous

```
public void autonomous()
```

Automated drive for autonomous mode. If in middle, drive forward, knock down bridge, turn around. Else, or then, go ahead and try to score.

**Overrides:**

`autonomous` in class `SimpleRobot`

## operatorControl

```
public void operatorControl()
```

Operator-controlled drive for Teleop mode. Handles robot driving, automated balancing for the bridge, ball pickup, turret aiming, firing, angle adjustments, light control, elevator control - both automated and manual - pneumatics, shifting, and various other things.

**Overrides:**

`operatorControl` in class `SimpleRobot`

## disabled

```
public void disabled()
```

The robot is disabled. Like ze goggles, zees does nothing.

**Overrides:**

`disabled` in class `SimpleRobot`

com._604robotics.robot2012.rotation

# Class SlowbroRotationProvider

java.lang.Object
  com._604robotics.robot2012.rotation.SlowbroRotationProvider

**All Implemented Interfaces:**

RotationProvider

---

```
public class SlowbroRotationProvider
extends Object
implements RotationProvider
```

Implements a slow-er-ish, but more robust-ish, RotationProvider.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **SlowbroRotationProvider**(**ConvertingPIDController** controller, **CameraInterface** cameraInterface, **Encoder** encoderTurret)<br>Initializes a new SlowbroRotationProvider. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **setDefaultPosition**(double defaultPosition)<br>Sets the "default" position, if no targets can be located. |
| boolean | **update**()<br>Updates the aiming of the turret. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### SlowbroRotationProvider

```
public SlowbroRotationProvider(ConvertingPIDController controller,
                               CameraInterface cameraInterface,
                               Encoder encoderTurret)
```

Initializes a new SlowbroRotationProvider.

**Parameters:**

`controller` - The PIDController to control.

`cameraInterface` - The CameraInterface to read data from.

`encoderTurret` - The turret encoder to read data from.

## Method Detail

### setDefaultPosition

```
public void setDefaultPosition(double defaultPosition)
```

**Description copied from interface:** `RotationProvider`
Sets the "default" position, if no targets can be located.

**Specified by:**

`setDefaultPosition` in interface `RotationProvider`

## update

```
public boolean update()
```

**Description copied from interface:** `RotationProvider`
Updates the aiming of the turret.

**Specified by:**

`update` in interface `RotationProvider`

com._604robotics.robot2012.rotation

# Class NaiveRotationProvider

java.lang.Object
  com._604robotics.robot2012.rotation.NaiveRotationProvider

**All Implemented Interfaces:**

RotationProvider

---

```
public class NaiveRotationProvider
extends Object
implements RotationProvider
```

A naive implementation of a RotationProvider,

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **NaiveRotationProvider**(**PIDController** controller, **CameraInterface** cameraInterface, **Encoder** encoderTurret)<br>Initializes a new NaiveRotationProvider, giving it control over the specified PIDController. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **setDefaultPosition**(double defaultPosition)<br>Sets the "default" position, if no targets can be located. |
| boolean | **update**()<br>U pdates the aiming of the turret. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### NaiveRotationProvider

```
public NaiveRotationProvider(PIDController controller,
                             CameraInterface cameraInterface,
                             E ncoderencoderTurret)
```

Initializes a new NaiveRotationProvider, giving it control over the specified PIDController.

**Parameters:**

`controller` - The PIDController to control.

`cameraInterface` - The CameraInterface to read data from.

`encoderTurret` - The turret encoder to read data from.

## Method Detail

### setDefaultPosition

```
public void setDefaultPosition(double defaultPosition)
```

**Description copied from interface:** `RotationProvider`
Sets the "default" position, if no targets can be located.

**Specified by:**

`setDefaultPosition` in interface `RotationProvider`

## update

```
public boolean update()
```

**Description copied from interface:** `RotationProvider`
Updates the aiming of the turret.

**Specified by:**

`update` in interface `RotationProvider`

com._604robotics.robot2012.rotation

# Class DummyRotationProvider

java.lang.Object
  com._604robotics.robot2012.rotation.DummyRotationProvider

**All Implemented Interfaces:**

RotationProvider

---

```
public class DummyRotationProvider
extends Object
implements RotationProvider
```

Dummy implementor of a RotationProvider, for testing purposes.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **DummyRotationProvider**(**PIDController** controller)<br>Initializes a new DummyRotationProvider, giving it control over the specified PIDController. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **setDefaultPosition**(double defaultPosition)<br>Sets the "default" position, if no targets can be located. |
| boolean | **update**()<br>U pdates the aimingof the turret. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### DummyRotationProvider

```
public DummyRotationProvider(PIDController controller)
```

Initializes a new DummyRotationProvider, giving it control over the specified PIDController.

**Parameters:**

controller - The PIDController to control.

## Method Detail

### setDefaultPosition

```
public void setDefaultPosition(double defaultPosition)
```

**Description copied from interface:** **RotationProvider**
Sets the "default" position, if no targets can be located.

**Specified by:**

`setDefaultPosition` in interface `RotationProvider`

## update

```
public boolean update()
```

**Description copied from interface: `RotationProvider`**

U pdates the aimingof the turret.

**Specified by:**

`update` in interface `RotationProvider`

com._604robotics.robot2012.rotation

# Interface RotationProvider

**All Known Implementing Classes:**

DummyRotationProvider, NaiveRotationProvider, SlightlySmarterRotationProvider, SlowbroRotationProvider

---

```
public interface RotationProvider
```

Based on external feedback, aims the turret at the target.

**Author:**

Michael Smith

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| void | **setDefaultPosition**(double defaultPosition)<br>Sets the "default" position, if no targets can be located. |
| boolean | **update**()<br>Updates the aiming of the turret. |

## Method Detail

### setDefaultPosition

```
void setDefaultPosition(double defaultPosition)
```

Sets the "default" position, if no targets can be located.

### update

```
boolean update()
```

Updates the aiming of the turret.

com._604robotics.robot2012.rotation

# Class SlightlySmarterRotationProvider

java.lang.Object
  com._604robotics.robot2012.rotation.SlightlySmarterRotationProvider

**All Implemented Interfaces:**

RotationProvider

---

```
public class SlightlySmarterRotationProvider
extends Object
implements RotationProvider
```

A slightly smarter implementation of a rotation provider, which tries to account for network delay, etc.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **SlightlySmarterRotationProvider**(PIDController controller, CameraInterface cameraInterface, Encoder encoderTurret)<br>Initializes a new SlightlySmarterRotationProvider. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **setDefaultPosition**(double defaultPosition)<br>Sets the "default" position, if no targets can be located. |
| boolean | **update**()<br>Updates the aiming of the turret. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### SlightlySmarterRotationProvider

```
public SlightlySmarterRotationProvider(PIDController controller,
                                       CameraInterface cameraInterface,
                                       Encoder encoderTurret)
```

Initializes a new SlightlySmarterRotationProvider.

**Parameters:**

`controller` - The PIDController to control.

`cameraInterface` - The CameraInterface to read data from.

`encoderTurret` - The turret encoder to read data from.

## Method Detail

### setDefaultPosition

```
public void setDefaultPosition(double defaultPosition)
```

**Description copied from interface:** `RotationProvider`
Sets the "default" position, if no targets can be located.

**Specified by:**

`setDefaultPosition` in interface `RotationProvider`

## update

```
public boolean update()
```

**Description copied from interface:** `RotationProvider`
Updates the aiming of the turret.

**Specified by:**

`update` in interface `RotationProvider`

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.RING_LIGHT

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.RING_LIGHT
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static Relay.Value | OFF |
| static Relay.Value | ON |

## Field Detail

### ON

```
static final Relay.Value ON
```

### OFF

```
static final Relay.Value OFF
```

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Encoders.Drive

**Enclosing interface:**

PortConfiguration.Encoders

---

```
public static interface PortConfiguration.Encoders.Drive
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | LEFT_A |
| static int | LEFT_B |
| static int | RIGHT_A |
| static int | RIGHT_B |

## Field Detail

### LEFT_A

```
static final int LEFT_A
```

**See Also:**

Constant Field Values

### LEFT_B

```
static final int LEFT_B
```

**See Also:**

Constant Field Values

### RIGHT_A

```
static final int RIGHT_A
```

**See Also:**

Constant Field Values

### RIGHT_B

```
static final int RIGHT_B
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.SOLENOID_SHOOTER

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.SOLENOID_SHOOTER
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static DoubleSolenoid.Value | LOWER_ANGLE |
| static DoubleSolenoid.Value | UPPER_ANGLE |

## Field Detail

### LOWER_ANGLE

```
static final DoubleSolenoid.Value LOWER_ANGLE
```

### UPPER_ANGLE

```
static final DoubleSolenoid.Value UPPER_ANGLE
```

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Relays

**Enclosing interface:**

PortConfiguration

---

```
public static interface PortConfiguration.Relays
```

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static Relay.Direction | RING_LIGHT_DIRECTION |
| static int | RING_LIGHT_PORT |

## Field Detail

### RING_LIGHT_PORT

```
static final int RING_LIGHT_PORT
```

**See Also:**

Constant Field Values

### RING_LIGHT_DIRECTION

```
static final Relay.Direction RING_LIGHT_DIRECTION
```

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.ELEVATOR.TOLERANCE

**Enclosing interface:**

ActuatorConfiguration.ELEVATOR

---

```
public static interface ActuatorConfiguration.ELEVATOR.TOLERANCE
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | HIGH |
| static int | LOW |
| static int | MEDIUM_LOWER |
| static int | MEDIUM_UPPER |

## Field Detail

### HIGH

```
static final int HIGH
```

**See Also:**

Constant Field Values

### MEDIUM_UPPER

```
static final int MEDIUM_UPPER
```

**See Also:**

Constant Field Values

### MEDIUM_LOWER

```
static final int MEDIUM_LOW
```

**See Also:**

Constant Field Values

### LOW

```
static final int LOW
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.ELEVATOR.DEADBAND

**Enclosing interface:**

ActuatorConfiguration.ELEVATOR

---

```
public static interface ActuatorConfiguration.ELEVATOR.DEADBAND
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | HIGH |
| static int | LOW |
| static int | MEDIUM_LOWER |
| static int | MEDIUM_UPPER |

## Field Detail

### HIGH

```
static final int HIGH
```

**See Also:**

Constant Field Values

### MEDIUM_UPPER

```
static final int MEDIUM_UPPER
```

**See Also:**

Constant Field Values

### MEDIUM_LOWER

```
static final int MEDIUM_LOW
```

**See Also:**

Constant Field Values

### LOW

```
static final int LOW
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Pneumatics.SHOOTER_SOLENOID

**Enclosing interface:**

PortConfiguration.Pneumatics

---

```
public static interface PortConfiguration.Pneumatics.SHOOTER_SOLENOID
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | LOWER_ANGLE |
| static int | UPPER_ANGLE |

## Field Detail

### LOWER_ANGLE

```
static final int LOWER_ANGLE
```

**See Also:**

Constant Field Values

### UPPER_ANGLE

```
static final int UPPER_ANGLE
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration

---

`public interface` **`PortConfiguration`**

Port configuration.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | **`PortConfiguration.Controllers`** |
| static interface | **`PortConfiguration.Encoders`** |
| static interface | **`PortConfiguration.Motors`** |
| static interface | **`PortConfiguration.Pneumatics`** |
| static interface | **`PortConfiguration.Relays`** |
| static interface | **`PortConfiguration.Sensors`** |

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Encoders

**Enclosing interface:**

PortConfiguration

---

```
public static interface PortConfiguration.Encoders
```

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | PortConfiguration.Encoders.Drive |

## S id Summary

**S ids**

| Modifier and Type | S id and Description |
|---|---|
| static int | ELEVATOR_A |
| static int | ELEVATOR_B |
| static int | TURRET_ROTATION_A |
| static int | TURRET_ROTATION_B |

## S id Detail

### ELEw F TA M D F

```
static final int S N S o _ , y z ) _
```

**See F lso**

Constant Field Values

### ELEw F TA M D G

```
static final int S N S o _ , y z ) v
```

**See F lso**

Constant Field Values

### TU M M ETD M A TF TIA ND F

```
static final int , q z z S , ) z y , _ , f y g ) _
```

**See F lso**

Constant Field Values

### TU M M ETD M A TF TIA ND G

```
static final int , q z z S , ) z y , _ , f y g ) v
```

**See F lso**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Controllers

**Enclosing interface:**

PortConfiguration

---

```
public static interface PortConfiguration.Controllers
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | DRIVE |
| static int | MANIPULATOR |

## Field Detail

### DRIVE

```
static final int DRO _ ,
```

**See Also:**

Constant Field Values

### MANIPULATOR

```
static final int MANO v q f z g Y N
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.SOLENOID_SHIFTER

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.SOLENOID_SHIFTER
```

## Field Summary

Field

| Modifier and Type | Field and Description |
|---|---|
| static DoubleSolenoid.Value | HIGH_GEAR |
| static DoubleSolenoid.Value | LOW_GEAR |

## Field Detail

### LOW_GEAR

```
static final DoubleSolenoid.Value LOW
```

### HIGH_GEAR

```
static final DoubleSolenoid.Value HIG
```

com._604robotics.robot2012.configuration

# Interface AutonomousConfiguration

public interface **AutonomousConfiguration**

Autonomous mode configuration.

**Author:**

Sebastian Merz , Michael Smith

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static S N uble | BACKWARD_DISTANCE |
| static S N uble | BACKWARD_DISTANCE_SIDES |
| static S N uble | BACKWARD_DRIVE_POWER |
| static S N uble | FORWARD_DISTANCE |
| static S N uble | FORWARD_DRIVE_POWER |

## Field Detail

### FORWARD_DISTANCE

static final S N uble o _ , y z , ) v ) q f g z Y h A

**See Also:**

Constant Field Values

### BACKWARD_DISTANCE

static final S N uble R z h N y z , ) v ) q f g z Y h A

**See Also:**

Constant Field Values

### BACKWARD_DISTANCE_SIDES

static final S N uble R z h N y z , ) v ) q f g z Y h A v f q ) A f

**See Also:**

Constant Field Values

### FORWARD_DRIVE_POWER

static final S N uble o _ , y z , ) v ) , q w A v R _ y A ,

**See Also:**

Constant Field Values

### BACKWARD_DRIVE_POWER

static final S N uble R z h N y z , ) v ) , q w A v R _ y A ,

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface SensorConfiguration

---

`public interface` **`SensorConfiguration`**

Sensor configuration.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| `static interface` | **`SensorConfiguration.Encoders`** |

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static S N uble` | **`ACCELEROMETER_SENSITIVITY`** |
| `static S N uble` | **`ACCELEROMETER_UPPER_RADIANS`** |
| `static S N uble` | **`GYRO_DRIFT`** |
| `static int` | **`TURRET_CALIBRATION_OFFSET`** |

## Field Detail

### GYRO_DRIFT

`static final S N uble o _ , y z ) , v q f`

**See Also:**

Constant Field Values

### ACCELEROMETER_SENSITIVITY

`static final S N uble g Y Y h A h , y R h f h , z N h w N v f v R v f _`

**See Also:**

Constant Field Values

### ACCELEROMETER_UPPER_RADIANS

`static final S N uble g Y Y h A h , y R h f h , z U N N h , z , g ) v g w N`

**See Also:**

Constant Field Values

### TURRET_CALIBRATION_OFFSET

`static final int f U , , h f z Y g A v X , g f v y w z y q q N h f`

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Motors

**Enclosing interface:**

PortConfiguration

---

public static interface **PortConfiguration.Motors**

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | ELEVATOR_LEFT |
| static int | ELEVATOR_RIGHT |
| static int | HOPPER |
| static int | LEFT_DRIVE |
| static int | PICKUP |
| static int | RIGHT_DRIVE |
| static int | SHOOTER_LEFT |
| static int | SHOOTER_RIGHT |
| static int | TURRET_ROTATION |

## Field Detail

### LEFT_DF IE

static final int LEFT_Dz ) v N

**See Also:**

Constant Field Values

### F IDTGDF IE

static final int z ) q f _ , y z ) v N

**See Also:**

Constant Field Values

### ELEA MRF wEWT

static final int ELEv g _ Y z , S N o _

**See Also:**

Constant Field Values

### ELEA MRF w F TD G

static final int ELEv g _ Y z , z ) q f _

**See Also:**

Constant Field Values

### SHR RET wEWT

static final int )Hv v zE, S N o _

**See Also:**

Constant Field Values

### SHR RETF w F TD G

`static final int )Hv v zE, z ) q f _`

**See Also:**

Constant Field Values

### HR PEF

`static final int Hv ) z E`

**See Also:**

Constant Field Values

### PICh V P

`static final int S) R N w A`

**See Also:**

Constant Field Values

### TVF IET_F TATIR N

`static final int Twz z N _ , z Y _ g _ ) Y R`

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ButtonConfiguration.Manipulator.Elevator

**Enclosing interface:**

ButtonConfiguration.Manipulator

---

```
public static interface ButtonConfiguration.Manipulator.Elevator
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | DOWN |
| static int | FORWARD |
| static int | LEFT |
| static int | RIGHT |

## Field Detail

### FORWARD

```
static final int S N o _ , o y
```

**See Also:**

Constant Field Values

### LEFT

```
static final int L) S v
```

**See Also:**

Constant Field Values

### RIGHT

```
static final int Rq f g v
```

**See Also:**

Constant Field Values

### DOWN

```
static final int DO_ Y
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Pneumatics.HOPPER_SOLENOID

**Enclosing interface:**

PortConfiguration.Pneumatics

---

```
public static interface PortConfiguration.Pneumatics.HOPPER_SOLENOID
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | FORWARD |
| static int | REVERSE |

## Field Detail

### FORWARD

```
static final int S N o _ , o y
```

**See Also:**

Constant Field Values

### REVERSE

```
static final int Rz ) z o v z
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface SensorConfiguration.Encoders

**Enclosing interface:**

SensorConfiguration

---

```
public static interface SensorConfiguration.Encoders
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static double | LEFT_DRIVE_INCHES_PER_CLICK |
| static double | RIGHT_DRIVE_INCHES_PER_CLICK |
| static double | TURRET_DEGREES_PER_CLICK |

## Field Detail

### TURRET_DEGREES_PER_CLICK

```
static final double o _ , , y o z ) y v , y y q z f y , z g Y h g A
```

**See Also:**

Constant Field Values

### LEFT_DRIVE_INCHES_PER_CLICK

```
static final double hy R o z ) , h N y z h w g R y q z f y , z g Y h g A
```

**See Also:**

Constant Field Values

### RIGHT_DRIVE_INCHES_PER_CLICK

```
static final double R) Gq z ) , h N y z h w g R y q z f y , z g Y h g A
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ButtonConfiguration.Manipulator

**Enclosing interface:**

ButtonConfiguration

---

public static interface **ButtonConfiguration.Manipulator**

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | ButtonConfiguration.Manipulator.Elevator |

## wield Summary

**wields**

| Modifier and Type | wield and Description |
|---|---|
| static int | AIM_AND_SHOOT |
| static int | PICKUP |
| static int | TOGGLE_ANGLE |
| static int | TOGGLE_HEIGHT |
| static int | TOGGLE_LIGHT |

## wield Detail

### w MF w NSAF MTM

static final int S N o _ S , y _ z ) v v q

**See w lso**

Constant Field Values

### PICD G P

static final int PIg h ) P

**See w lso**

Constant Field Values

### TM U EFL ÆIRAT

static final int TOGGV v ) N N A ) q

**See w lso**

Constant Field Values

### TM U EFLw NE L

static final int TOGGV v S , A R N

**See w lso**

Constant Field Values

TM U UEFLL IUT A

```
static final int TOGGV v R N A ) q
```

**See w Iso**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Pneumatics

**Enclosing interface:**

PortConfiguration

---

```
public static interface PortConfiguration.Pneumatics
```

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | PortConfiguration.Pneumatics.HOPPER_SOLENOID |
| static interface | PortConfiguration.Pneumatics.PICKUP_SOLENOID |
| static interface | PortConfiguration.Pneumatics.SHIFTER_SOLENOID |
| static interface | PortConfiguration.Pneumatics.SHOOTER_SOLENOID |

## S ield Summary

**S ields**

| Modifier and Type | S ield and Description |
|---|---|
| static int | COMPRESSOR |
| static int | PRESSURE_SWITCH |

## S ield Detail

### Cw MPVESSw w

```
static final int S N o _ , y z z N ,
```

**See Also:**

Constant Field Values

### PwESSUwEMSD ITCG

```
static final int PRy z z ) , y v z q f g S Y
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.SOLENOID_HOPPER

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.SOLENOID_HOPPER
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static DoubleSolenoid.Value | PUSH |
| static DoubleSolenoid.Value | REGULAR |

## Field Detail

### REGULAR

```
static final DoubleSolenoid.Value z ) v q f g z
```

### PUSH

```
static final DoubleSolenoid.Value h ⊟H
```

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.SOLENOID_PICKUP

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.SOLENOID_PICKUP
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static DoubleSolenoid.Value | IN |
| static DoubleSolenoid.Value | OUT |

## Field Detail

### IN

```
static final DoubleSolenoid.Value z )
```

### OUT

```
static final DoubleSolenoid.Value OUf
```

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.TURRET_POSITION

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.TURRET_POSITION
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static double | FORWARD |
| static double | LEFT |
| static double | RIGHT |
| static double | TOLERANCE |

## Field Detail

### FORWARD

```
static final double F_ , y z , )
```

**See Also:**

Constant Field Values

### LEFT

```
static final double v q o f
```

**See Also:**

Constant Field Values

### RIGHT

```
static final double Rg hf )
```

**See Also:**

Constant Field Values

### TOLERANCE

```
static final double f _ v q , z A R q
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration

---

```
public interface ActuatorConfiguration
```

Actuator polarity and power configuration.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | ActuatorConfiguration.ELEVATOR |
| static interface | ActuatorConfiguration.RING_LIGHT |
| static interface | ActuatorConfiguration.SOLENOID_HOPPER |
| static interface | ActuatorConfiguration.SOLENOID_PICKUP |
| static interface | ActuatorConfiguration.SOLENOID_SHIFTER |
| static interface | ActuatorConfiguration.SOLENOID_SHOOTER |
| static interface | ActuatorConfiguration.TURRET_POSITION |

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static S N uble | ACCELEROMETER_DRIVE_POWER |
| static S N uble | ELEVATOR_POWER_MAX |
| static S N uble | ELEVATOR_POWER_MIN |
| static S N uble | HOPPER_POWER |
| static S N uble | HOPPER_POWER_REVERSE |
| static S N uble | PICKUP_POWER |
| static S N uble | TURRET_ROTATION_POWER_MAX |
| static S N uble | TURRET_ROTATION_POWER_MIN |

## Field Detail

### ACCELEROMETER_DRIVE_POWER

```
static final S N uble o _ _ , y , z ) v , q , z f g z Y h , f A ) R , z
```

**See Also:**

Constant Field Values

### HOPPER_POWER

```
static final S N uble N ) A A , z f A ) R , z
```

**See Also:**

Constant Field Values

### HOPPER_POWER_REVERSE

```
static final S N uble N ) A A , z f A ) R , z f z , h , z w ,
```

**See Also:**

Constant Field Values

### PICKUP_POWER

`static final S N uble A Y _ R U A f A ) R , z`

**See Also:**

Constant Field Values

### ELEVATOR_POWER_MIN

`static final S N uble , y , h o q ) z f A ) R , z f v Y N`

**See Also:**

Constant Field Values

### ELEVATOR_POWER_MAX

`static final S N uble , y , h o q ) z f A ) R , z f v o X`

**See Also:**

Constant Field Values

### TURRET_ROTATION_POWER_MIN

`static final S N uble q U z z , q f z ) q o q Y ) N f A ) R , z f v Y N`

**See Also:**

Constant Field Values

### TURRET_ROTATION_POWER_MAX

`static final S N uble q U z z , q f z ) q o q Y ) N f A ) R , z f v o X`

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Sensors

**Enclosing interface:**

PortConfiguration

---

```
public static interface PortConfiguration.Sensors
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | ACCELEROMETER |
| static int | ELEVATOR_LIMIT_SWITCH |
| static int | GYRO_BALANCE |
| static int | GYRO_HEADING |

## Field Detail

### w w F ÆRMDIN w

```
static final int GNo_, yz)vqf S
```

**See R Iso**

Constant Field Values

### w w F A M U E L G NC

```
static final int GNo_, g) Y) f h z
```

**See R Iso**

Constant Field Values

### R CCEhEF AMETEF

```
static final int A) ȥ Y z o _ A z R z o
```

**See R Iso**

Constant Field Values

### EhEV RATF NMLTMSW TICD

```
static final int z Y z N) R_o, Y q A q R, w R q R h y
```

**See R Iso**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Pneumatics.PICKUP_SOLENOID

**Enclosing interface:**

PortConfiguration.Pneumatics

---

```
public static interface PortConfiguration.Pneumatics.PICKUP_SOLENOID
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | IN |
| static int | OUT |

## Field Detail

### IN

```
static final int S N
```

**See** D **Iso**

Constant Field Values

### OUA

```
static final int O_ ,
```

**See** D **Iso**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ButtonConfiguration.Driver

**Enclosing interface:**

ButtonConfiguration

---

```
public static interface ButtonConfiguration.Driver
```

## Field Summary

Fields

| Modifier and Syw e | Field and Descriw ibn |
|---|---|
| static int | AUTO_BALANCE |
| static int | GYRO_RESET |
| static int | SHIFT |
| static int | TOGGLE_PICKUP |

## Field Detail

### Sw FIS

```
static final int SHo _ ,
```

**See Also:**

Constant Field Values

### S A MEVR PICR h P

```
static final int , y z z ) v q f o g Y h f
```

**See Also:**

Constant Field Values

### Ah S ABQAD F MEC

```
static final int S ) y q R A ) A N g v
```

**See Also:**

Constant Field Values

### M Y R ÆESQESR

```
static final int GS R q R v S v ,
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface PortConfiguration.Pneumatics.SHIFTER_SOLENOID

**Enclosing interface:**

PortConfiguration.Pneumatics

---

```
public static interface PortConfiguration.Pneumatics.SHIFTER_SOLENOID
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | HIGH_GEAR |
| static int | LOW_GEAR |

## Field Detail

### LOW_GEAR

```
static final int LOW_ , y z )
```

**See Also:**

Constant Field Values

### HIGH_GEAR

```
static final int Hq , v _ , y z )
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.configuration

# Interface ButtonConfiguration

---

`public interface` **`ButtonConfiguration`**

Button configuration.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| `static interface` | **`ButtonConfiguration.Driver`** |
| `static interface` | **`ButtonConfiguration.Manipulator`** |

com._604robotics.robot2012.configuration

# Interface ActuatorConfiguration.ELEVATOR

**Enclosing interface:**

ActuatorConfiguration

---

```
public static interface ActuatorConfiguration.ELEVATOR
```

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | ActuatorConfiguration.ELEVATOR.DEADBAND |
| static interface | ActuatorConfiguration.ELEVATOR.TOLERANCE |

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static int | HIGH |
| static int | LOW |
| static int | MEDIUM |
| static int | OKAY_TO_TURN |

## Field Detail

### HIGH

```
static final int HIGH
```

**See Also:**

Constant Field Values

### MEDIUM

```
static final int MEDIUM
```

**See Also:**

Constant Field Values

### LOW

```
static final int LOW
```

**See Also:**

Constant Field Values

### OKAY_TO_TURN

```
static final int OKAY_TO_TURN
```

**See Also:**

Constant Field Values

com._604robotics.robot2012.vision

# Class Point3d

java.lang.Object
    com._604robotics.robot2012.vision.Point3d

---

```
public class Point3d
extends Object
```

This represents a point in 3d space

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| double | **x**<br>the x value |
| double | **y**<br>the y value |
| double | **z**<br>the z value |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **Point3d**(double x, double y, double z) |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| double | **getX**() |
| double | **getY**() |
| double | **getZ**() |
| void | **setX**(double x)<br>Sets the X value of this Point |
| void | **setY**(double y)<br>Sets the Y value of this Point |
| void | **setZ**(double z)<br>Sets the Z value of this Point |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### x

```
public double x
```

the x value

### y

```
public double y
```

the y value

### z

```
public double z
```

the z value

## Constructor Detail

### Point3d

```
public Point3d(double x,
       double y,
       double z)
```

**Parameters:**

`x` - - the x value

`y` - - the y value

`z` - - the z value

## Method Detail

### getX

```
public double getX()
```

**Returns:**

- the X value

### setX

```
public void setX(double x)
```

Sets the X value of this Point

**Parameters:**

`x` - - the X value

### getY

```
public double getY()
```

**Returns:**

- the Y value

### setY

```
public void setY(double y)
```

Sets the Y value of this Point

**Parameters:**

`y` - - the Y value

### getZ

```
public double getA()
```

**Returns:**

- the ( value

### setZ

```
public void setZ(double z)
```

Sets the ( value ozthis Point

**Parameters:**

`z` - - the ( value

com._604robotics.robot2012.vision

# Class Target

java.lang.Object
    com._604robotics.robot2012.vision.Target

---

```
public class Target
extends Object
```

Represents a target.

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| double | **angle**<br>This is the angle of the targetErelative to the camera. |
| double | **angle_uncertainty**<br>This is the uncertainty of the angle of the target. |
| double | **x**<br>xEyEand z represent the W ( pbsition of the target x will be positive when the target appears to be right of the center of the camera. |
| double | **x_uncertainty**<br>These are the uncertainties of the xEyEand z positions of the target. |
| double | **y**<br>xEyEand z represent the W ( pbsition of the target x will be positive when the target appears to be right of the center of the camera. |
| double | **y_uncertainty**<br>These are the uncertainties of the xEyEand z positions of the target. |
| double | **z**<br>xEyEand z represent the W ( pbsition of the target x will be positive when the target appears to be right of the center of the camera. |
| double | **z_uncertainty**<br>These are the uncertainties of the xEyEand z positions of the target. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **Target**() |
| **Target**(double x, double y, double z, double angle) |
| **Target**(double x, double y, double z, double x_uncertainty, double y_uncertainty, double z_uncertainty, double angle, double angle_uncertainty) |
| **Target**(**Point3d** point, double angle) |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| **String** | **toString**() |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Field Detail

### angle

```
public double angle
```

This is the angle of the target relative to the camera.

) angle1
.....) Target1
.....2
.....2
.....2
....2( ( ( ( ( ( |! ) Camera1
...2
..2
.2
2
this value is expressed in radians.

### angle_uncertainty

```
public double angle_uncertainty
```

This is the uncertainty of the angle of the target. This is interpreted as a plus or minus to the angle. Again this is expressed in radians

### x

```
public double x
```

x y and z represent the W ( position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative ) see ikipediaRight( hand rule1 As the absolute value of z increases so does the distance from the camera to the target. To determine the approximate accuracy of these values check - x y z > _accuracy The units of these measures are in inches.

### y

```
public double y
```

x y and z represent the W ( position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative ) see ikipediaRight( hand rule1 As the absolute value of z increases so does the distance from the camera to the target. To determine the approximate accuracy of these values check - x y z > _accuracy The units of these measures are in inches.

### z

```
public double z
```

x y and z represent the W ( position of the target x will be positive when the target appears to be right of the center of the camera. y will be positive when the target appears to be above of the center of the camera. z will always be negative ) see ikipediaRight( hand rule1 As the absolute value of z increases so does the distance from the camera to the target. To determine the approximate accuracy of these values check - x y z > _accuracy The units of these measures are in inches.

### x_uncertainty

```
public double x_uncertainty
```

These are the uncertainties of the x y and z positions of the target. These are interpreted as pluses and minuses to the x y and z values. Again these are in inches.

### y_uncertainty

```
public double y_uncertainty
```

These are the uncertainties of the x y and z positions of the target. These are interpreted as pluses and minuses to the x y and z values. Again these are in inches.

### z_uncertainty

```
public double z_uncertainty
```

These are the uncertainties of the x y and z positions of the target. These are interpreted as pluses and minuses to the x y and z values. Again these are in inches.

## Constructor Detail

### Target

```
public Warget(double x,
      double y,
      double z,
      double angle)
```

**Parameters:**

`x`(

`y`(

`z`(

`angle`(

### Target

```
public Target(double x,
      double y,
      double z,
      double x_uncertainty,
      double y_uncertainty,
      double z_uncertainty,
      double angle,
      double angle_uncertainty)
```

**Parameters:**

`x`(

`y`(

`z`(

`x_uncertainty`(

`y_uncertainty`(

`z_uncertainty`(

`angle`(

`angle_uncertainty`(

### Target

```
public Target(Point3d point,
      double angle)
```

**Parameters:**

`point`(

`angle`(

### Target

```
public Target()
```

## Method Detail

### toString

```
public String toString()
```

**Overrides:**

`toString` in class `Object`

com._604robotics.robot2012.aiming

# Class Point3d

java.lang.Object
    com._604robotics.robot2012.aiming.Point3d

---

```
public class Point3d
extends Object
```

Represents a single point in 3D space.

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| double | **x** |
| double | **y** |
| double | **z** |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **Point3d**()<br>InitialiE es a new Poir3d. |
| **Point3d**(double x, double y, double z)<br>InitialiE es a new Poir3d. |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getg ass, hashCode, notify, notify) ll, toStV ng, wait, wait, wait

## Field Detail

### x

```
public double x
```

### y

```
public double y
```

### z

```
public double z
```

## Constructor Detail

### Point3d

```
public Point3d()
```

Initializes a new Point3d.

### Point3d

```
public Point3d(double x,
               double y,
               double z)
```

Initializes a new Point3d.

**Parameters:**

- `x` The x coordinate of the point.
- `y` The y coordinate of the point.
- `z` The z coordinate of the point.

com._604robotics.robot2012.aiming

# Class Point2d

java.lang.Object
    com._604robotics.robot2012.aiming.Point2d

---

```
public class Point2d
extends Object
```

Represents a single point on the 2D plane.

**Author:**

Kevin parker

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **Point2d**(double x, double y)<br>Intializes a new Point2d. |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### Point2d

```
public Point2d(double x,
       double y)
```

Intializes a new Point2d.

**Parameters:**

x – The xE coordinate oz the point.

y – The yE coordinate oz the point.

com._604robotics.robot2012.aiming

# Class PointAndAngle3d

java.lang.Object
    com._604robotics.robot2012.aiming.PointAndAngle3d

---

```
public class PointAndAngle3d
extends Object
```

A class to hold a 3d point.

**Author:**

Kevin Parker , Sebastian Merz

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **PointAndAngle3d**(double x, double y, double z, double angle)<br>Initializes variables for the point. |
| **PointAndAngle3d**(**Point3d** p, double angle)<br>Initializes variables for the point. |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### PointAndAngle3d

```
public Point3d(double x,
               double y,
               double z,
               double angle)
```

Initializes variables for the point.

**Parameters:**

x The x coordinate of the point.

y The y coordinate of the point.

z The z coordinate of the point.

angle The angle the target is at from the robot.

### PointAndAngle3d

```
public Point3d(Point3d p,
               double angle)
```

Initializes variables for the point.

**Parameters:**

p Uses the values from this point to create the new point.

angle Uses this angle for the new point.

com._604robotics.robot2012.aiming

# Class Aiming

java.lang.Object
    com._604robotics.robot2012.aiming.Aiming

---

```
public class Aiming
extends Object
```

Utility class for various aiming functions and such.

**Author:**

Kevin Parker

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static Aiming | defaultAiming |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| Aiming() |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| PointAndAngle3d | getAngleAndRelXYZOfTarget(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4)<br>E etheangle from the targets, and the relative distances of the corners of the target as perceived by the camera. |
| double | getAngleOfTarget(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4, double z)<br>This function gets the direction the target is facing, relative to the camera. |
| Point3d | getRelXYZOfTarget(double x1, double y1, double w, double h)<br>W emembethat this re( uiresthe camera to be ) pefectly) flat, and the targets to be ) pefectly) vertical. |
| Point3d | getRelXYZOfTarget(Target t) |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### defaultAiming

```
public static final Aiming defaultAiming
```

## Constructor Detail

### Aiming

**Aiming**

```
public q iynW()
```

## Method Detail

### getRelXYZOfTarget

```
public Aoint3d getRelXYZOfTarget(double x1,
                         double y1,
                         double w,
                         double h)
```

W emembethat this re( uiresthe camera to be ) pefectly) flat, and the targets to be ) pefectly) vertical. A new function will probably need to be created for use on the robot. That, or we'll need to manipulate the points based on camera angle. The points are in the following pattern: 2 y | 1 2 | | - 4 2 - - - - - - - > 2 x

**Parameters:**

`x1` - x- valueof the bottom left corner

`y1` - y- valueof the bottom left corner

`w` - width of the vision target

`h` - height of the vision target

**Returns:**

a Point- dholding the X Y zand Z of the target, relative to the camera.

### getRelXYZOfTarget

```
public Point3d getRelXYZOfTarget(Target t)
```

### getAngleOfTarget

```
public double getAngleOfTarget(double x1,
                       double y1,
                       double x2,
                       double y2,
                       double x3,
                       double y3,
                       double x4,
                       double y4,
                       double z)
```

This function gets the direction the target is facing, relative to the camera. It is imperfect, and half- assumesa simple orthographic projection ( whichis not ( uitelike real life) If it causes issues ( whichthe accuracy of this function doesn't need to be very high) we can fix it later.

**Parameters:**

`x1` - x- valueof the bottom left corner

`y1` - y- valueof the bottom left corner

`x2` -

`y2` -

`x3` -

`y3` -

`x4` -

`y4` -

`z` -

**Returns:**

the resulting angle in radians.

### getAngleAndRelXYZOfTarget

```
public PointAndAngle3d getAngleAndRelXYZOfTarget(double x1,
                                         double y1,
                                         double x2,
                                         double y2,
                                         double x3,
                                         double y3,
```

```
                                    double x4,
                                    double y4)
```

Get the angle from the targets, and the relative distances of the corners of the target as perceived by the camera.

**Parameters:**

x1 -

y1 -

x2 -

y2 -

x3 -

y3 -

x4 -

y4 -

**Returns:**

com._604robotics.robot2012.camera

# Interface CameraInterface

**All Known Implementing Classes:**

RemoteCameraTCP

---

```
public interface CameraInterface
```

Represents a method for obtaining processed vision data from the camera.

**Author:**

Michael Smith

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| void | **begin**()<br>Launches the CameraInterface. |
| void | **end**()<br>Disables the CameraInterface. |
| double | **getRecordedTime**()<br>Gets the estimated time since the last packet was received. |
| **Target**[] | **getTargets**()<br>Returns the most recently-obtained array of Target that represents the visible targets. |

## Method Detail

### begin

```
void begin()
```

Launches the CameraInterface.

### end

```
void end()
```

Disables the CameraInterface.

### getTargets

```
Target[] getTargets()
```

Returns the most recently-obtained array of Target that represents the visible targets.

**Returns:**

An array of Target that represents the visible targets.

### getRecordedTime

```
double getRecordedTime()
```

Gets the estimated time since the last packet was received.

**Returns:**

The estimated time since the last packet was received.

com._604robotics.robot2012.camera

# Class RemoteCameraTCP

java.lang.Object
  com._604robotics.robot2012.camera.RemoteCameraTCP

**All Implemented Interfaces:**

CameraInterface

---

```
public class RemoteCameraTCP
extends Object
implements CameraInterface
```

Implements a CameraInterface that draws data from a TCP connection.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| RemoteCameraTCP() |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | begin() Initializes communication. |
| void | end() E nds communication. |
| double | getRecordedTime() Records the time elapsed between reception of data packets from camera. |
| Target[] | getTargets() Returns the last targets acz uired from the remote software. |
| int | getUPu() Returns the number of updates received per second. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### RemoteCameraTCP

```
public RemoteCameraTCP()
```

## Method Detail

### begin

```
public void begin()
```

Initializes communication.

**Specified by:**

beV in interface CameraInterface

### end

```
public void end()
```

E ndscommunication.

**Specified by:**

end in interface CameraInterface

### getTargets

```
public Target[] getTargets()
```

Returns the last targets acz uiredfrom the remote software.

**Specified by:**

getTargets in interface CameraInterface

**Returns:**

The last targets acz uiredfrom the remote software.

### getRecordedTime

```
public double getRecordedTime()
```

Records the time elapsed between reception of data packets from camera.

**Specified by:**

getRecordedTime in interface CameraInterface

**Returns:**

The elapsed time since the last packet was received.

### getUPS

```
public int getUPS()
```

Returns the number of updates received per second. For testing and debugging purposes.

**Returns:**

The number of updates per second.

com._604robotics.utils

# Class SpringableRelay

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Relay
      com._604robotics.utils.SpringableRelay

**All Implemented Interfaces:**

IDevice, IDeviceController

---

```
public class SpringableRelay
extends Relay
```

Extender of a Relay providing an easier control flow. When an output is set for the Relay, it is considered "sprung". When the "reload" method is called, if the victor is sprung, it unsprings the Relay. If the Relay is not sprung, then the output is set to the default output. In this way, the Relay will only be moving when you tell it to. Use this in a loop or something, and call "reload" at the end. No more worries about code paths that don't update the Relays!

**Author:**

Michael Smith

## Nested Class Summary

### Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.Relay

Relay.Direction, Relay.InvalidValueException, Relay.Value

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **SpringableRelay**(int moduleNumber, int channel, **Relay.Direction** direction, **Relay.Value** defaultDirection)<br>Initializes a new SpringableRelay. |
| **SpringableRelay**(int moduleNumber, int channel, **Relay.Value** defaultDirection)<br>Initializes a new SpringableRelay. |
| **SpringableRelay**(int channel, **Relay.Direction** direction, **Relay.Value** defaultDirection)<br>Initializes a new SpringableRelay. |
| **SpringableRelay**(int channel, **Relay.Value** defaultDirection)<br>Initializes a new SpringableRelay. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| boolean | **getSprung**()<br>Has the Relay been sprung? |
| void | **reload**()<br>If the Relay has been sprung, unspring it; if not, set the output to the default output. |
| void | **set**(**Relay.Value** direction)<br>Sets the direction of the Relay. |

| | |
|---|---|
| void | **spring**()<br>Springs the Relay. |

### Methods inherited from class edu.wpi.first.wpilibj.Relay

free, setDirection

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### SpringableRelay

```
public SpringableRelay(int moduleNumber,
                       int channel,
                       Relay.Direction direction,
                       Relay.Value defaultDirection)
```

Initializes a new SpringableRelay.

**Parameters:**

`moduleNumber` - The module slot the Relay is on.

`channel` - The channel the Relay is on.

`direction` - The direction the Relay should control.

`defaultDirection` - The default direction for reloading.

### SpringableRelay

```
public SpringableRelay(int channel,
                       Relay.Direction direction,
                       Relay.Value defaultDirection)
```

Initializes a new SpringableRelay.

**Parameters:**

`channel` - The channel the Relay is on.

`direction` - The direction the Relay should control.

`defaultDirection` - The default direction for reloading.

### SpringableRelay

```
public SpringableRelay(int moduleNumber,
                       int channel,
                       Relay.Value defaultDirection)
```

Initializes a new SpringableRelay.

**Parameters:**

`moduleNumber` - The module slot the Relay is on.

`channel` - The channel the Relay is on.

`defaultDirection` - The default direction for reloading.

### SpringableRelay

```
public SpringableRelay(int channel,
                       Relay.Value defaultDirection)
```

Initializes a new SpringableRelay.

**Parameters:**

**Parameters:**

`cVannel` - The channel theRelay is on.

`def aultDirection` - The default direction for reloading.

## Method Detail

### getSprung

`public boolean getSprung()`

Has the Relay been sprung2

**Returns:**

E hether or not theRelay has been sprung.

### spring

`public void spring()`

Springs the Relay.

### set

`public void set(Relay.Value direction)`

Sets the direction of the Relay.

**Overrides:**

`set` in class `Relay`

**Parameters:**

`direction` - The direction to set.

### reload

`public void reload()`

If the Relay has been sprung, unspring it! if not, set the output to the defaultoutput.

com._604robotics.utils

# Class CompensatingGyro

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Gyro
      edu.wpi.first.wpilibj.GyroHax
        com._604robotics.utils.CompensatingGyro

**All Implemented Interfaces:**

IDevice, ISensor, PIDSource

---

```
public class CompensatingGyro
extends GyroHax
```

Gyro with manual compensation-setting support.

**Author:**

Michael Smith

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **CompensatingGyro**(**AnalogChannel** channel)<br>InitialiE esa new CompensatingGyro on the specified AnalogChannel. |
| **CompensatingGyro**(int port)<br>InitialiE esa new CompensatingGyro on the specified Pz Mport. |
| **CompensatingGyro**(int slot, int port)<br>InitialiE esa new CompensatingGyro on the specified Pz Mport on the specified module port. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **setAccumulatorCenter**(int center)<br>Manually sets the center for the accumulator. |

### Methods inherited from class edu.wpi.first.wpilibj.GyroHax

getAnalogChannel

### Methods inherited from class edu.wpi.first.wpilibj.Gyro

free, getAngle, pidGet, reset, setSensitivity

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### CompensatingGyro

```
public CompensatingGyro(int port)
```

InitialiE esa new CompensatingGyro on the specified Pz Mport. Note that port must be Wor ( )

**Parameters:**

`port` - The Pz Mport the gyro is plugged into. Must be Wor ( )

### CompensatingGyro

```
public CompensatingGyro(int slot,
                        int port)
```

InitialiE esa new CompensatingGyro on the specified Pz Mport on the specified module port. Note that port must be Wor ( )

**Parameters:**

`slot` - The module slot the gyro is plugged into.

`port` - The Pz Mport the gyro is plugged into. Must be Wor ( )

### CompensatingGyro

```
public CompensatingGyro(AnalogChannel channel)
```

InitialiE esa new CompensatingGyro on the specified AnalogChannel. Note that port must be Wor ( )

**Parameters:**

`channel` - The AnalogChannel the gyro is plugged into.

## Method Detail

### setAccumulatorCenter

```
public void setAccumulatorCenter(int center)
```

Manually sets the center for the accumulator.

**Parameters:**

`center` - The center to set.

com._604robotics.utils

# Class DeadbandedSource

java.lang.Object
    com._604robotics.utils.DeadbandedSource

**All Implemented Interfaces:**

PIDSource

---

```
public class DeadbandedSource
extends Object
implements PIDSource
```

Implements a PIDSource, wrapping around another PIDSource, with a deadband range. If we're within the deadband, it'll tell the PIDController we're at where it wants to be.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **DeadbandedSource**(PIDSource source)<br>Initializes a new DeadbandedSource. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| double | **pidGet**()<br>Hooks into PIDSource - gets the value to send to the PIDController. |
| goid | **setController**(PIDController controller)<br>Sets the PIDController the source is fed into. |
| goid | **setDeadband**(double lo herDeadband) double upperDeadband)<br>Sets the range for the deadband. |

### Methods inherited from class java.lang.Object

clone) eS als) V ialive) wetq ass) zasz qde) notiV )C notiV C y) lfoStrinW) hait) hait) hait

## Constructor Detail

### DeadbandedSource

```
public DeadbandedSource(PIDSource source)
```

Initializes a new DeadbandedSource.

**Parameters:**

source - The underlying PIDSource to wrap around.

## Method Detail

### setController

```
public goid setController(PIDController controller)
```

Sets the PIDController the source is fed into.

**Parameters:**

`controller` - The PIDController the source is fed into.

## setDeadband

```
public goid setDeadband(double loherDeadband)
                        double upperDeadband)
```

Sets the range for the deadband.

**Parameters:**

`loherDeadband` - The lower bound of the deadband.

`upperDeadband` - The upper bound of the deadband.

## pidGet

```
public double pidAet()
```

Hooks into PIDSource - gets the value to send to the PIDController. E  ith a deadbandz

**Specified by:**

`pidAet` in interface `PIDSource`

**Returns:**

The value to send to the PIDController.

com._604robotics.utils

# Class UpDownPIDController

java.lang.Object
  edu.wpi.first.wpilibj.PIDController
    com._604robotics.utils.UpDownPIDController

**All Implemented Interfaces:**

IDevice, IUtility

---

```
public class UpDownPIDController
extends PIDController
```

A PIDController with different gains for up and down.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static class | **UpDownPIDController.Gains**<br>A structure containing the P, I, and D gains. |

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.PIDController

kDefaultPeriod

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **UpDownPIDController**(**UpDownPIDController.Gains** upGains, **UpDownPIDController.Gains** downGains, **PIDSource** source, **PIDOutput** output)<br>Initializes a new UpDownPIDController. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| **UpDownPIDController.Gains** | **getDownGains**()<br>Gets the Gains for going down. |
| **UpDownPIDController.Gains** | **getUpGains**()<br>Gets the Gains for going up. |
| void | **refreshGains**()<br>Updates the gains for the current direction. |
| void | **setDownGains**(**UpDownPIDController.Gains** downGains)<br>Sets the gains for going down. |
| void | **setSetpoint**(double setpoint)<br>Sets the setpoint to go to. |
| void | **setUpGains**(**UpDownPIDController.Gains** upGains)<br>Sets the gains for going up. |

### Methods inherited from class edu.wpi.first.wpilibj.PIDController

disable, enable, free, get, getD, getError, getI, getP, getSetpoint, isEnable, onTarget, reset, setContinuous, setContinuous, setInputRange, setOutputRange, setPID, setTolerance

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### UpDownPIDController

```
public UpDownPIDController(UpDownPIDController.Gains upGains,
                   UpDownPIDController.Gains downGains,
                   PIDSource source,
                   PIDOutput output)
```

Initializes a new UpDownPIDController.

**Parameters:**

`upGains` E The gains to use when going up.

`downGains` E The gains to use when going down.

`source` E The PIDSource to plug in.

`output` E The PIDOutput to plug in.

## Method Detail

### getUpGains

```
public UpDownPIDController.Gains getUpGains()
```

Gets the Gains for going up.

**Returns:**

The gains for going up.

### getDownGains

```
public UpDownPIDController.Gains getDownGains()
```

Gets the Gains for going down.

**Returns:**

The gains for going down.

### refreshGains

```
public void refreshGains()
```

Updates the gains for the current direction.

### setUpGains

```
public void setUpGains(UpDownPIDController.Gains upGains)
```

Sets the gains for going up.

**Parameters:**

`upGains` E The gains to use when going up.

### setDownGains

```
public void setDownGains(UpDownPIDController.Gains downGains)
```

Sets the gains for going down.

Sets the gains for going down.

**Parameters:**

`downGains` E The gains to use when going down.

## setSetpoint

```
public void setSetpoint(double setpoint)
```

Sets the setpoint to go to.

**Overrides:**

`setSetpoint` in class `PIDController`

**Parameters:**

`setpoint` E The setpoint to go to.

com._604robotics.utils

# Class DualVictor

java.lang.Object
    com._604robotics.utils.DualVictor

**All Implemented Interfaces:**

PIDOutput

---

```
public class DualVictor
extends Object
implements PIDOutput
```

Control two Victors like they're one. Useful for PID controllers. Also, it's springable E see SpringableVictorz .

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **DualVictor**(int leftPo rt, int rig htPo rt)<br>InitialiWe a DualVictor with a left and a right P( M port. |
| **DualVictor**(int leftS ot, int leftPo rt, int rig htS ot, int rig htPo rt)<br>InitialiWes a DualVictor with left and right slot and P( M port. |
| **DualVictor**(**Victor** leftV ito r **Victor** r igtW ito r )<br>InitialiWes a DualVictor with left and right slot and P( M port. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| do ubè | **get**()<br>Checks the current power the Victors are set to. |
| bo oean | **getSprung**()<br>Has the victor been sprung) |
| vo id | **pidWrite**(do ubè o tput)<br>Function to hook into the PIDController. |
| vo id | **reload**()<br>If the Victor has been sprung, unspring it1 finot, set the output to 0. |
| vo id | **set**(do ubè speed)<br>Sets the power of the Victors. |
| vo id | **setController**(**PIDController** co ntroller)<br>Sets the PIDController for this DualVictor, if there is one. |
| void | **setDeadband**(double lowerDeadband, double upperDeadband)<br>Sets the deadband for the DualVictor. |
| void | **setLeftInversion**(boolean inVersion)<br>Sets the inversion for the " left" Victor. |
| void | **setRightInversion**(boolean inVersion)<br>Sets the inversion for the " right" Victor. |
| void | **setSafetyEnabled**(boolean enabled)<br>Sets whether or not safety is enabled. |
| void | **spring**()<br>Springs the victor. |

### Methods inherited from class java.lang.Object

clone, eq als, finaliZe, getC ass, hashCode, notify, notify A ,lltoString, wait, wait, wait

## Constructor Detail

### DualVictor

```
public DualVictor(int leftPort,
          int rightPort)
```

Initialize a DualVictor with a left and a right PWM port.

**Parameters:**

`leftPort` - The PWM port for the 2 left Victor.

`rightPort` - The PWM port for the 2 right Victor.

### DualVictor

```
public DualVictor(int leftSlot,
          int leftPort,
          int rightSlot,
          int rightPort)
```

Initializes a DualVictor with left and right slot and PWM port.

**Parameters:**

`leftSlot` - The slot the left Victor is plugged into.

`leftPort` - The PWM port for the 2 left Victor.

`rightSlot` - The slot the "right" Victor is plugged into.

`rightPort` - The PWM port for the 2 right Victor.

### DualVictor

```
public DualVictor(Victor leftVictor,
          Victor rightVictor)
```

Initializes a DualVictor with left and right slot and PWM port.

**Parameters:**

`leftVictor` - The left Victor.

`rightVictor` - The "right" Victor.

## Method Detail

### getSprung

```
public boolean getSprung()
```

Has the victor been sprung)

**Returns:**

(whether or not the victor has been sprung.

### spring

```
public void spring()
```

Springs the victor.

### setLeftInversion

```
public void setLeftInversion(boolean inversion)
```

Sets the inversion for the 2 left Victor.

**Parameters:**

`inversion` - Is it inverted)

### setRightInversion

```
public void setRightInversion(boolean inversion)
```

Sets the inversion for the 2 right Victor.

**Parameters:**

`inAersion` - Is it inverted

## get

```
public double get()
```

Checks the current power the Victors are set to.

**Returns:**

The current power the Victors are set to.

## set

```
public Aoid set(double speed)
```

Sets the power of the Victors.

**Parameters:**

`speed` - The speed to set.

## pidWrite

```
public Aoid pidWrite(double output)
```

Function to hook into the PIDController. Sets the power of the Victors.

**Specified by:**

`pidWrite` in interface `PIDOutput`

**Parameters:**

`output` - The speed to set.

## setDeadband

```
public Aoid setDeadband(double lowerDeadband,
                double upperDeadband)
```

Sets the deadband for the DualVictor. The default is no deadband.

**Parameters:**

`lowerDeadband` - The lower bound of the deadband.

`upperDeadband` - The upper bound of the deadband.

## setSafetyEnabled

```
public Aoid setSafet( Enabled(boolean enabled)
```

Sets whether or not safety is enabled.

**Parameters:**

`enabled` - ( hether or not safety is enabled.

## reload

```
public Aoid reload()
```

If the Victor has been sprung, unspring it1 finot, set the output to 0.

## setController

```
public Aoid setJontroller(PIDJontroller controller)
```

Sets the PIDController for this DualVictor, if there is one. If the PIDController is enabled, reload will assume it's updating it, and won't reset the output to 0.

**Parameters:**

`controller`! The PIDControllerfor this DualVictor.

com._604robotics.utils

# Class SpringableVictor

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.PWM
      edu.wpi.first.wpilibj.SafePWM
        edu.wpi.first.wpilibj.Victor
          com._604robotics.utils.SpringableVictor

**All Implemented Interfaces:**

MotorSafety, IDevice, IDeviceController, PIDOutput, SpeedController

---

```
public class SpringableVictor
extends Victor
```

E xtender of a Victor providing an easier control flow. When an output is set for the Victor, it is considered z sprungz When the z reloadz method is called, if the victor is sprung, it unsprings the Victor. If the Victor is not sprung, then the output is set to zero. In this way, the Victor will only be moving when you tell it to. ( se this in a loop or something, and call z reloadz at the end. No more worries about code paths that don) t update the Victors!

**Author:**

Michael Smith

## Nested Class Summary

### Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.PWM

PWM.PeriodMultiplier

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.PWM

kDefaultMinPwmHigh, kDefaultPwmPeriod, kPwmDisabled

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

### Fields inherited from interface edu.wpi.first.wpilibj.MotorSafety

DEFAULT_SAFETY_EXPIRATION

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **SpringableVictor**(int port)<br>Initializes a new SpringableVictor on the given PWM port. |
| **SpringableVictor**(int slot, int port)<br>Initializes a new SpringableVictor on the given module slot and PWM port. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| boolean | getSprung() |

| | |
|---|---|
| boolean | **getSprung**( )<br>Has the victor been sprung2 |
| void | **pidWrite**( double output)<br>Function to hook into the PIDController. |
| void | **reload**( )<br>If the Victor has been sprung, unspring it! if notset the output to 0. |
| void | **set**( double speed)<br>Sets the power of the Victor. |
| void | **setController**( **PIDController** controller)<br>Sets the PIDController for this Victor, if there is one. |
| void | **spring**()<br>Springs the victor. |

### Methods inherited from class edu.wpi.first.wpilibj.Victor

get, set

### Methods inherited from class edu.wpi.first.wpilibj.SafePWM

disable, Feed, getDescription, getExpiration, isAlive, isSafetyEnabled, setExpiration, setSafetyEnabled, stopMotor

### Methods inherited from class edu.wpi.first.wpilibj.PWM

enableDeadbandElimination, free, getChannel, getModuleNumber, getPosition, getRaw, getSpeed, setBounds, setPeriodMultiplier, setPosition, setRaw

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Methods inherited from interface edu.wpi.first.wpilibj.SpeedController

disable

## Constructor Detail

### SpringableVictor

```
public SpringableVictor(int port)
```

InitialiWes a new SpringableVictor on the given PWM port.

**Parameters:**

`port` - The PWM port the Victor is connected to.

### SpringableVictor

```
public SpringableVictor(int slot,
                        int port)
```

Initializes a new SpringableVictor on the given module slot and PWM port.

**Parameters:**

`slot` - The module slot theVictor is connected to.

`port` - The PWM port the Victor is connected to.

## Method Detail

### getSprung

### getSprung

```
public boolean getSprung()
```

Has the victor been sprung2

O **eturns:**

Whether or not the victor has been sprung.

### spring

```
public void spring()
```

Springs the victor.

### set

```
public void set(double speed)
```

Sets the power of the Victor.

**Specified by:**

`set` in interface `SpeedController`

O **verdes:**

`set` in class `Victor`

**Parameters:**

`speed` - The speed to set.

### pidF ite

```
public void pidWrite(double output)
```

Function to hook into the PIDController. Sets the power of the Victors.

**Specified by:**

`pidWrite` in interface `PIDOutput`

O **verdes:**

`pidWrite` in class `Victor`

**Parameters:**

`output` - The speed to set.

### reload

```
public void reload()
```

If the Victor has been sprung, unspring it! if notset the output to 0.

### setController

```
public void setController(PIDController controller)
```

Sets the PIDController for this Victor, if there is one. If the PIDController is enabled, reload will assume it) s updating jtand won) t reset the output to 0.

**Parameters:**

`controller` - The PIDController for this Victor.

com._604robotics.utils

# Class ConvertingPIDController

java.lang.Object
  edu.wpi.first.wpilibj.PIDController
    com._604robotics.utils.ConvertingPIDController

**All Implemented Interfaces:**

IDevice, IUtility

---

```
public class ConvertingPIDController
extends PIDController
```

An extender of a PIDController that converts between units when getting and setting a setpoint.

**Author:**

Michael Smith

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.PIDController

kDefaultPeriod

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **ConvertingPIDController**(double Kp, double Ki, double Kd, **PIDSource** source, **PIDOutput** output)<br>Allocate a PID object with the given constants for P, I, D, using a 50ms period. |
| **ConvertingPIDController**(double Kp, double Ki, double Kd, **PIDSource** source, **PIDOutput** output, double period)<br>Allocate a PID object with the given constants for P, I, D |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| double | **getRealSetpoint**()<br>Gets the "real" setpoint of the PIDController. |
| double | **getSetpoint**()<br>Returns the current setpoint of the PIDController |
| void | **setConversionFactor**(double conversionFactor)<br>Sets the factor to use when doing conversion on setSetpoint and getSetpoint. |
| void | **setRealSetpoint**(double setpoint)<br>Sets the "real" setpoint of the PIDController. |
| void | **setSetpoint**(double setpoint)<br>Set the setpoint for the PIDController |

### Methods inherited from class edu.wpi.first.wpilibj.PIDController

disable, enable, free, get, getD, getError, getI, getP, isEnable, onTarget, reset, setContinuous, setContinuous, setInputRange, setOutputRange, setPID, setTolerance

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### ConvertingPIDController

```
public ConvertingPIDController(double Kp,
                        double Ki,
                        double Kd,
                        PIDSource source,
                        PIDOutput output)
```

Allocate a PID object with the given constants for P, I, D, using a 50ms period.

**Parameters:**

`Kp` - the proportional coefficient

`Ki` - the integral coefficient

`Kd` - the derivative coefficient

`source` - The PIDSource object that is used to get values

`output` - The PIDOutput object that is set to the output value

### ConvertingPIDController

```
public ConvertingPIDController(double Kp,
                        double Ki,
                        double Kd,
                        PIDSource source,
                        PIDOutput output,
                        double period)
```

Allocate a PID object with the given constants for P, I, D

**Parameters:**

`Kp` - the proportional coefficient

`Ki` - the integral coefficient

`Kd` - the derivative coefficient

`source` - The PIDSource object that is used to get values

`output` - The PIDOutput object that is set to the output value

`period` - the loop time for doing calculations. This particularly effects calculations of the integral and differential terms. The default is 50ms.

## Method Detail

### getRealSetpoint

```
public double getRealSetpoint()
```

Gets the "real" setpoint of the PIDController.

**Returns:**

The "real" setpoint of the PIDController.

### getSetpoint

```
public double getSetpoint()
```

**Description copied from class:** `edu.wpi.first.wpilibj.PIDController`
Returns the current setpoint of the PIDController

**Overrides:**

`getSetpoint` in class `PIDController`

**Returns:**

the current setpoint

### setRealSetpoint

```
public void setRealSetpoint(double setpoint)
```

Sets the "real" setpoint of the PIDController

Sets the E realE  setpoint of the PIDController.

**Parameters:**

`setpoint` W The E realE  setpoint to set.

## setSetpoint

```
public void setSetpoint(double setpoint)
```

**Description copied from class:** `edu.wpi.first.wpilibj.PIDController`
Set the setpoint for the PIDController

**Overrides:**

`setSetpoint` in class `PIDController`

**Parameters:**

`setpoint` W the desired setpoint

## setConversionFactor

```
public void setConversionFactor(double conversionFactor)
```

Sets the factor to use when doing conversion on setSetpoint and getSetpoint.

**Parameters:**

`conversionFactor` W The conversion factor to use.

com._604robotics.utils

# Class LinearController

java.lang.Object
    com._604robotics.utils.LinearController

---

```
public class LinearController
extends Object
```

This class implements a controller with a horizontal segment, a linear segment, and finally a coasting segment. When a target point is set, the controller decides which direction to go to get there, and then focuses on getting to that point or past it in that direction. If that condition is met, the output drops to zero. E lse, if we are within a certain coasting range, the output will be floored at the coasting output. E lse, if we are outside a certain horizontal range, the output will be ceilinged at a certain horizontal output. E lse, the output will be scaled linearly between the two outputs.

**Author:**

Michael Smith

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **LinearController**(**PIDSource** source, **PIDOutput** output, double horizontalRange, double horizontalOutput, double coastingRange, double coastingOutput)<br>Initializes a new LinearController. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| double | **calculate**()<br>Function that performs the output calculation. |
| double | **getTarget**()<br>( ets the current target. |
| boolean | **onTarget**()<br>Are we there yet) |
| void | **setCoastingRange**(double coastingRange, double coastingOutput)<br>1 pdates the coasting values. |
| void | **setHorizontalRange**(double horizontalRange, double horizontalOutput)<br>1 pdates the horizontal values. |
| void | **setTarget**(double target)<br>Sets the current target. |
| void | **update**()<br>1 pdates the PIDOutput based on the latest data. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### LinearController

```
public LinearController(PIDSource source,
                        PIDOutput output,
                        double horizontalRange,
                        double horizontalOutput,
                        double coastingRange,
                        double coastingOutput)
```

Initializes a new LinearController.

**Parameters:**

`source` – A PIDSource to read from.

`output` – A PIDOutput to write to.

`horizontalRange` – The horizontal range, as defined in the class description.

`horizontalOutput` – The horizontal output, as defined in the class description.

`coastingRange` – The coasting range, as defined in the class description.

`coastingOutput` – The coasting output, as defined in the class description.

## Method Detail

### setHorizontalRange

```
public void setHorizontalRange(double horizontalRange,
                               double horizontalOutput)
```

Updates the horizontal values.

**Parameters:**

`horizontalRange` – The horizontal range, as defined in the class description.

`horizontalOutput` – The horizontal output, as defined in the class description.

### setCoastingRange

```
public void setCoastingRange(double coastingRange,
                             double coastingOutput)
```

Updates the coasting values.

**Parameters:**

`coastingRange` – The coasting range, as defined in the class description.

`coastingOutput` – The coasting output, as defined in the class description.

### getTarget

```
public double getTarget()
```

Gets the current target.

**Returns:**

The current target.

### setTarget

```
public void setTarget(double target)
```

Sets the current target.

**Parameters:**

`target` – The target to move toward.

### onTarget

```
public boolean onTarget()
```

Are we there yet?

**Returns:**

Whether or not we're there yet.

### calculate

```
public double calculate()
```

Function that performs the output calculation. Exposed for debug use, mainly.

**Returns:**

**Returns:**

An output value, to be passed to a PIDOutput.

### update

```
public Aoid update()
```

U pdates the PIDOutput based on the latest data.

com._604robotics.utils

# Interface XboxController.Axis

**Enclosing class:**

XboxController

---

`public static interface` **`XboxController.Axis`**

Enumeration for the available axes on the Xbox controller.

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **`LEFT_STICK_X`** |
| `static int` | **`LEFT_STICK_Y`** |
| `static int` | **`RIGHT_STICK_X`** |
| `static int` | **`RIGHT_STICK_Y`** |

## Field Detail

### LEFT_STICK_X

`static final int LEFT_STICK_X`

**See Also:**

Constant Field Values

### LEFT_STICK_Y

`static final int LEFT_STICK_Y`

**See Also:**

Constant Field Values

### RIGHT_STICK_X

`static final int RIGHT_STICK_X`

**See Also:**

Constant Field Values

### RIGHT_STICK_Y

`static final int RIGHT_STICK_Y`

**See Also:**

Constant Field Values

com._604robotics.utils

# Class EncoderPIDSource

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Encoder
      com._604robotics.utils.EncoderOffset
        com._604robotics.utils.EncoderPIDSource

**All Implemented Interfaces:**

CounterBase, IDevice, ISensor, PIDSource

---

```
public class EncoderPIDSource
extends EncoderOffset
```

Encoder extender that return the value of Encoder.get() when pidGet is called. Drop-in replacement: all constructors from the Encoder class are implemented here.

**Author:**

Michael Smith

## Nested Class Summary

### Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.Encoder

Encoder.PIDSourceParameter

### Nested classes/interfaces inherited from interface edu.wpi.first.wpilibj.CounterBase

CounterBase.EncodingType

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.Encoder

m_aSource, m_bSource, m_indexSource

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

Constructors

| Constructor and Description |
| --- |
| **EncoderPIDSource**(**DigitalSource** aSource, **DigitalSource** bSource)<br>Encoder constructor. |
| **EncoderPIDSource**(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection)<br>Encoder constructor. |
| **EncoderPIDSource**(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection, **CounterBase.EncodingType** encodingType)<br>Encoder constructor. |
| **EncoderPIDSource**(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource)<br>Encoder constructor. |
| **EncoderPIDSource**(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource, boolean reverseDirection)<br>Encoder constructor. |
| **EncoderPIDSource**(int aChannel, int bChannel)<br>Encoder constructor. |
| **EncoderPIDSource**(int aChannel, int bChannel, boolean reverseDirection) |

Encoder constructor.

**EncoderPIDSource**(int aChannel, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)
Encoder constructor.

**EncoderPIDSource**(int aChannel, int bChannel, int indexChannel)
Encoder constructor.

**EncoderPIDSource**(int aChannel, int bChannel, int indexChannel, boolean reverseDirection)
Encoder constructor.

**EncoderPIDSource**(int aSlot, int aChannel, int bSlot, int bChannel)
Encoder constructor.

**EncoderPIDSource**(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection)
Encoder constructor.

**EncoderPIDSource**(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)
Encoder constructor.

**EncoderPIDSource**(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel)
Encoder constructor.

**EncoderPIDSource**(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel, boolean reverseDirection)
Encoder constructor.

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| double | **pidGet**()<br>Hooks into the PIDSource interface. |

### Methods inherited from class com._604robotics.utils.EncoderOffset

getRaw, reset, setOffset

### Methods inherited from class edu.wpi.first.wpilibj.Encoder

free, get, getDirection, getDistance, getPeriod, getRate, getStopped, setDistancePerPulse, setMaxPeriod, setMinRate, setPIDSourceParameter, setReverseDirection, start, stop

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` The a channel digital input module.

`aChannel` The a channel digital input channel.

`bSlot` The b channel digital input module.

`bChannel` The b channel digital input channel.

`reverseDirection` represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

## EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k( ) , k1 ) , or k4) to indicate ( ) , decoding 4) If 4) is selected, then an encoder FPz object is used and the returned counts will be 4x the encoder spec2 d value since all rising and falling edges are counted. If ( ) or 1are selected then a counter object will be used and the returned value will either exactly match the spec2 d count or be double (2x) the spec'd count.

## EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                int indexSlot,
                int indexChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. Using the index pulse forces 4x encoding.

**Parameters:**

`aSlot` - The a channel digital input module.

`aChannel` - The a channel digital input channel.

`bSlot` - The b channel digital input module.

`bChannel` - The b channel digital input channel.

`indexSlot` - The index channel digital input module.

`indexChannel` - The index channel digital input channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderPIDSource

```
public EncoderPIDSource(int aSlot,
                int aChannel,
                int bSlot,
                int bChannel,
                int indexSlot,
                int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. Using the index pulse forces 4x encoding.

**Parameters:**

`aSlot` - The a channel digital input module.

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`indexSlot` W The index channel digital input module.

`indexChannel` W The index channel digital input channel.

## EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

## EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k( ) , k1 ) , or k4) to indicate ( ) , decoding4) If 4) is selected, then an encoder FPz dObject is used and the returned counts will be 4x the encoder spec2 d value since alrising and falling edges are counted. If ( ) or 1ar) e selected then a counter object will be used and the returned value will either exactly match the spec2 d count or be double(2x) the spec'd count.

## EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
                int bChannel,
                int indexChannel,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. Using an index pulse forces 4x encoding

**Parameters:**

`aChannel` - The a channel digital input channel.

`bChannel` - The b channel digital input channel.

`indexChannel` - The index channel digital input channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderPIDSource

```
public EncoderPIDSource(int aChannel,
```

```
                int bChannel,
                int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. ! sing an index pulse forces 4x encodig

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`indexChannel` W The index channel digital input channel.

## EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

## EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                boolean reverseDirection,
                CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k( ) , k1 ) , or k4) to indicate ( ) , decoding4) If 4) is selected, then an encoder FPz object is used and the returned counts will be 4x the encoder spec2 d value since all rising and falling edges are counted. If ( ) or 1are selected then a counter object will be used and the returned value will either exactly match the spec2 d count or be double(2x) the spec'd count.

## EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                DigitalSource bSource,
                DigitalSource indexSource,
                boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` - The source that should be used for the a channel.

`bSource` - the source that should be used for the b channel.

`indexSource` - the source that should be used for the index channel.

`reverseDirection` - represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

### EncoderPIDSource

```
public EncoderPIDSource(DigitalSource aSource,
                        DigitalSource bSource,
                        DigitalSource indexSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`indexSource` W the source that should be used for the index channel.

## Method Detail

### pidGet

```
public double pidGet()
```

Hooks into the PIDSource interface. This method overrides the one implemented by the underlying Encoder class, simply returning the value of this.get();

**Specified by:**

`pidGet` in interface `PIDSource`

**Overrides:**

`pidGet` in class `Encoder`

**Returns:**

The value to pass back to the PIDSource; in this case, that of this.get();

com._604robotics.utils

# Class EncoderOffset

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Encoder
      com._604robotics.utils.EncoderOffset

**All Implemented Interfaces:**

CounterBase, IDevice, ISensor, PIDSource

**Direct Known Subclasses:**

EncoderPIDSource

---

```
public class EncoderOffset
extends Encoder
```

Encoder extender that return the value of Encoder.get() when pidGet is called. Drop-in replacement: all constructors from the Encoder class are implemented here.

**Author:**

Michael Smith

## Nested Class Summary

### Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.Encoder

Encoder.PIDSourceParameter

### Nested classes/interfaces inherited from interface edu.wpi.first.wpilibj.CounterBase

CounterBase.EncodingType

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.Encoder

m_aSource, m_bSource, m_indexSource

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

Constructors

| Constructor and Description |
| --- |
| **EncoderOffset**(**DigitalSource** aSource, **DigitalSource** bSource)<br>Encoder constructor. |
| **EncoderOffset**(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection)<br>Encoder constructor. |
| **EncoderOffset**(**DigitalSource** aSource, **DigitalSource** bSource, boolean reverseDirection, **CounterBase.EncodingType** encodingType)<br>Encoder constructor. |
| **EncoderOffset**(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource)<br>Encoder constructor. |
| **EncoderOffset**(**DigitalSource** aSource, **DigitalSource** bSource, **DigitalSource** indexSource, boolean reverseDirection)<br>Encoder constructor. |
| **EncoderOffset**(int aChannel, int bChannel) |

Encoder constructor.

| | |
|---|---|
| **EncoderOffset**(int aChannel, int bChannel, boolean reverseDirection)<br>Encoder constructor. | |
| **EncoderOffset**(int aChannel, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)<br>Encoder constructor. | |
| **EncoderOffset**(int aChannel, int bChannel, int indexChannel)<br>Encoder constructor. | |
| **EncoderOffset**(int aChannel, int bChannel, int indexChannel, boolean reverseDirection)<br>Encoder constructor. | |
| **EncoderOffset**(int aSlot, int aChannel, int bSlot, int bChannel)<br>Encoder constructor. | |
| **EncoderOffset**(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection)<br>Encoder constructor. | |
| **EncoderOffset**(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType)<br>Encoder constructor. | |
| **EncoderOffset**(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel)<br>Encoder constructor. | |
| **EncoderOffset**(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel, boolean reverseDirection)<br>Encoder constructor. | |

## J ethod Summary

Methods

| Modifier and Type | Method and Description |
|---|---|
| int | **getRaw**()<br>z ets the raw value from the encoder. |
| void | **reset**()<br>( esets the Encoder. |
| void | **setOffset**(int offset)<br>Sets the offset value for the Encoder. |

### Methods inherited from class edu.wpi.first.wpilibj.Encoder

free, get, getDirection, getDistance, getPeriod, getRate, getStopped, pidGet, setDistancePerPulse, setMaxPeriod, setMinRate, setPIDSourceParameter, setReverseDirection, start, stop

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### EncoderOffset

```
public EncoderOffset(int aSlot,
             int aChannel,
             int bSlot,
             int bChannel,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderOffset

```
public EncoderOffset(int aSlot,
             int aChannel,
             int bSlot,
             int bChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

## EncoderOffset

```
public EncoderOffset(int aSlot,
             int aChannel,
             int bSlot,
             int bChannel,
             boolean reverseDirection,
             CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k) 1 , k2 1 , or k41 to indicate ) 1 , decoding If 41 is selected, then an encoder FPz object is used and the returned counts will be 4x the encoder spec! d value since all rising and falling edges are counted. If ) 1 or 2 are selected then a counter object will be used and the returned value will either exactly match the spec! d count or be double(2 xE the spec! d count.

## EncoderOffset

```
public EncoderOffset(int aSlot,
             int aChannel,
             int bSlot,
             int bChannel,
             int indexSlot,
             int indexChannel,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. - sing the index pulse forces 4x encoding.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`indexSlot` W The index channel digital input module.

`indexChannel` W The index channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderOffset

```
public EncoderOffset(int aSlot,
             int aChannel,
             int bSlot,
             int bChannel,
             int indexSlot,
             int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b modules and channels fully specified. - sing the index pulse forces 4x encoding.

**Parameters:**

`aSlot` W The a channel digital input module.

`aChannel` W The a channel digital input channel.

`bSlot` W The b channel digital input module.

`bChannel` W The b channel digital input channel.

`indexSlot` W The index channel digital input module.

`indexChannel` W The index channel digital input channel.

## EncoderOffset

```
public EncoderOffset(int aChannel,
             int bChannel,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderOffset

```
public EncoderOffset(int aChannel,
             int bChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

## EncoderOffset

```
public EncoderOffset(int aChannel,
             int bChannel,
             boolean reverseDirection,
             CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module.

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k) 1 , k2 1 , or k41 to indicate ) 1 , 2x decoding or 4x decoding. If k41 is selected, then an encoder FPz object is used and the returned counts will be 4x the encoder spec! d value since all rising and falling edges are counted. If ) 1 or 2x are selected then a counter object will be used and the returned value will either exactly match the spec! d count or be double (2 xE the spec! d count.

## EncoderOffset

```
public EncoderOffset(int aChannel,
             int bChannel,
             int indexChannel,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. - sing an index pulse forces 4x encoding

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`indexChannel` W The index channel digital input channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderOffset

## EncoderOffset

```
public EncoderOffset(int aChannel,
             int bChannel,
             int indexChannel)
```

Encoder constructor. Construct a Encoder given a and b channels assuming the default module. - sing an index pulse forces 4x encodig

**Parameters:**

`aChannel` W The a channel digital input channel.

`bChannel` W The b channel digital input channel.

`indexChannel` W The index channel digital input channel.

## EncoderOffset

```
public EncoderOffset(DigitalSource aSource,
             DigitalSource bSource,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

## EncoderOffset

```
public EncoderOffset(DigitalSource aSource,
             DigitalSource bSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

## EncoderOffset

```
public EncoderOffset(DigitalSource aSource,
             DigitalSource bSource,
             boolean reverseDirection,
             CounterBase.EncodingType encodingType)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k) 1 , k2 1 , or k41 to indicate ) 1 , decoding41 If 41 is selected, then an encoder FPz object is used and the returned counts will be 4x the encoder spec! d value since all rising and falling edges are counted. If ) 1 or 2are selected then a counter object will be used and the returned value will either exactly match the spec! d count or be double(2 xE the spec! d count.

## EncoderOffset

```
public EncoderOffset(DigitalSource aSource,
             DigitalSource bSource,
             DigitalSource indexSource,
             boolean reverseDirection)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`indexSource` W the source that should be used for the index channel.

`reverseDirection` W represents the orientation of the encoder and inverts the output values if necessary so forward represents positive values.

### EncoderOffset

```
public EncoderOffset(DigitalSource aSource,
             DigitalSource bSource,
             DigitalSource indexSource)
```

Encoder constructor. Construct a Encoder given a and b channels as digital inputs. This is used in the case where the digital inputs are shared. The Encoder class will not allocate the digital inputs and assume that they already are counted.

**Parameters:**

`aSource` W The source that should be used for the a channel.

`bSource` W the source that should be used for the b channel

`indexSource` W the source that should be used for the index channel.

## Method Detail

### getRaw

```
public int getRaw()
```

**Description copied from class:** `edu.wpi.first.wpilibj.Encoder`

z ets the raw value from the encoder. The raw value is the actual count unscaled by the ) x, 2 x, or 4x scale factor.

**Overrides:**

`getRaw` in class `Encoder`

**Returns:**

Current raw count from the encoder

### reset

```
public void reset()
```

( esets the Encoder. Also undoes any offsets previously set.

**Specified by:**

`reset` in interface `CounterBase`

**Overrides:**

`reset` in class `Encoder`

### setOffset

```
public void setOffset(int offset)
```

Sets the offset value for the Encoder.

**Parameters:**

`offset` W The offset value for the encoder.

com._604robotics.utils

# Class UpDownPIDController.Gains

java.lang.Object
    com._604robotics.utils.UpDownPIDController.Gains

**Enclosing class:**

UpDownPIDController

---

```
public static class UpDownPIDController.Gains
extends Object
```

A structure containing the P, I, and D gains.

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| double | D |
| double | I |
| double | P |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| UpDownPIDController.Gains(double P, double I, double D) |

## Method Summary

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Field Detail

### P

```
public double P
```

### I

```
public double I
```

### D

```
public double D
```

## Constructor Detail

### UpDownPIDController.Gains

```
public J poXnPIDController.Gains(double P,
                                 double I,
                                 double D)
```

com._604robotics.utils

# Class XboxController

java.lang.Object
    com._604robotics.utils.XboxController

---

```
public class XboxController
extends Object
```

Wrapper joystick class for the Xbox 360 controllers.

**Author:**

Michael Smith

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Class and Description |
|---|---|
| static interface | **XboxController.Axis**<br>Enumeration for the available axes on the Xbox controller. |
| static interface | **XboxController.Button**<br>Enumeration for the available buttons on the Xbox controller. |
| static interface | **XboxController.Stick**<br>Enumeration for the available sticks on the Xbox controller. |

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **XboxController**(int port)<br>Initialize a new XboxController on the specified port. |
| **XboxController**(**Joystick** joystick)<br>Initialize a new XboxController from the underlying Joystick. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| double | **getAxis**(int axis)<br>Get the value of the specified axis. |
| boolean | **getButton**(int button)<br>Get whether or not the specified button is currently pressed. |
| **Joystick** | **getJoystick**()<br>Gets the underlying Joystick object. |
| boolean | **getStick**(int stick)<br>Get whether or not there's a value reading on the stick. |
| boolean | **getToggle**(int button)<br>Get the toggle state of the specified button. |
| void | **resetToggles**()<br>Resets the toggle registry for the contrller. |
| void | **setD eabdnd**(int axis, double lower, double upper)<br>Sets the deadband for a particular axis. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### XboxController

`public XboxController(int port)`

Initialize a new XboxController on the specified port.

**Parameters:**

`port` - The USB port the controller is connected to.

### XboxController

`public XboxController(Joystick joystick)`

Initialize a new XboxController from the underlying Joystick.

**Parameters:**

`joystick` - The Joystick to overlay the XboxController interface on.

## Method Detail

### getAxis

`public double getAxis(int axis)`

Get the value of the specified axis.

**Parameters:**

`axis` - One of the axis values specified in XboxController.Axis.

### getStick

`public boolean getStick(int stick)`

Get whether or not there's a value reading on the stick.

**Parameters:**

`stick` - One of the stick values specified in XboxController.Stick.

**Returns:**

Whether or not there's a value reading on the stick.

### getButton

`public boolean getButton(int button)`

Get whether or not the specified button is currently pressed.

**Parameters:**

`button` - One of the button values specified in XboxController.Button.

### resetToggles

`public void resetToggles()`

Resets the toggle registry for the contrller.

### getToggle

`public boolean getToggle(int button)`

Get the toggle state of the specified button.

**Parameters:**

`button` - One of the button values specified in XboxController.Button.

### getJoysticA

```
public Joystick getJoystick()
```

Gets the underlying Joystick object. What, is XboxController not good enough for you?

M **eturns:**

The underlying Joystick object.

### setDeadband

```
public void setDeadband(int axis,
                        double lower,
                        double upper)
```

Sets the deadband for a particular axis.

**Parameters:**

`axis` - The axis to set the deadband for.

`lower` - The lower bound of the deadband.

`upper` - The upper bound of the deadband.

com._604robotics.utils

# Interface XboxController.Stick

**Enclosing class:**

XboxController

---

`public static interface` **`XboxController.Stick`**

Enumeration for the available sticks on the Xbox controller.

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| `static int` | **`DPAD`** |
| `static int` | **`LEFT_STICK`** |
| `static int` | **`RIGHT_STICK`** |

## Field Detail

### LEFT_STICK

`static final int LEFT_STICK`

**See Also:**

Constant Field Values

### RIGHT_STICK

`static final int RIGA TSTICK`

**See Also:**

Constant Field Values

### DPAD

`static final int DPAD`

**See Also:**

Constant Field Values

com._604robotics.utils

# Interface XboxController.Button

**Enclosing class:**

XboxController

---

```
public static interface XboxController.Button
```

Enumeration for the available buttons on the Xbox controller.

## Nested Class Summary

**Nested Classes**

| Modifier and Type | Interface and Description |
|---|---|
| static interface | XboxController.Button.DPad |

## Field Summary

**Fields**

| Modifier and Type | Field and Description |
|---|---|
| static int | A |
| static int | B |
| static int | Back |
| static int | EitherTrigger |
| static int | LB |
| static int | LeftStick |
| static int | LT |
| static int | RB |
| static int | RightStick |
| static int | RT |
| static int | Start |
| static int | X |
| static int | Y |

## Field Detail

### A

```
static final int A
```

**See** A **Iso**

Constant Field Values

### B

```
static final int B
```

**See** A **Iso**

Constant Field Values

### X

```
static final int X
```

**See** A **Iso**

See Also:

Constant Field Values

### M

```
static final int Y
```

**See Also:**

Constant Field Values

### DB

```
static final int LB
```

**See Also:**

Constant Field Values

### RB

```
static final int RB
```

**See Also:**

Constant Field Values

### Back

```
static final int Back
```

**See Also:**

Constant Field Values

### Start

```
static final int Start
```

**See Also:**

Constant Field Values

### D eftStickR

```
static final int LeftStick
```

**See Also:**

Constant Field Values

### Righ tStickR

```
static final int RightStick
```

**See Also:**

Constant Field Values

### D S

```
static final int LT
```

**See Also:**

Constant Field Values

### RT

```
static final int RT
```

**See** A **Iso**

Constant Field Values

### Eith eTrigger

```
static final int EitherTrigger
```

**See** A **Iso**

Constant Field Values

com._604robotics.utils

# Class SpringableDoubleSolenoid

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.SolenoidBase
      edu.wpi.first.wpilibj.DoubleSolenoid
        com._604robotics.utils.SpringableDoubleSolenoid

**All Implemented Interfaces:**

IDevice, IDeviceController

---

```
public class SpringableDoubleSolenoid
extends DoubleSolenoid
```

Extender of a DoubleSolenoid providing an easier control flow. When an output is set for the DoubleSolenoid, it is considered "sprung". When the "reload" method is called, if the victor is sprung, it unsprings the DoubleSolenoid. If the DoubleSolenoid is not sprung, then the output is set to the default output. In this way, the DoubleSolenoid will only be moving when you tell it to. Use this in a loop or something, and call "reload" at the end. No more worries about code paths that don't update the DoubleSolenoids!

**Author:**

Michael Smith

## Nested Class Summary

### Nested classes/interfaces inherited from class edu.wpi.first.wpilibj.DoubleSolenoid

DoubleSolenoid.Value

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.SolenoidBase

m_allocated, m_moduleNumber

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

Constructors

| Constructor and Description |
| --- |
| **SpringableDoubleSolenoid**(int forwardChannel, int reverseChannel, **DoubleSolenoid.Value** defaultDirection)<br>Initializes a new SpringableDoubleSolenoid. |
| **SpringableDoubleSolenoid**(int moduleNumber, int forwardChannel, int reverseChannel, **DoubleSolenoid.Value** defaultDirection)<br>Initializes a new SpringableDoubleSolenoid. |

## Method Summary

Methods

| Modifier and Type | Method and Description |
| --- | --- |
| boolean | **getSprung**()<br>Has the DoubleSolenoid been sprung? |
| void | **reload**()<br>If the DoubleSolenoid has been sprung, unspring it; if not, set the output to the default output. |

| | |
|---|---|
| void | **set**(**DoubleSolenoid.Value** direction)<br>Sets the direction of the DoubleSolenoid. |
| void | **spring**( )<br>Springs the DoubleSolenoid. |

### Methods inherited from class edu.wpi.first.wpilibj.DoubleSolenoid

free, get

### Methods inherited from class edu.wpi.first.wpilibj.SolenoidBase

getAll, getAllFromDefaultModule, getAllFromModule, set

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### SpringableDoubleSolenoid

```
public SpringableDoubleSolenoid(int forwardChannel,
                        int reverseChannel,
                        DoubleSolenoid.Value defaultDirection)
```

Initializes a new SpringableDoubleSolenoid.

**Parameters:**

`forwardChannel` - The forward channel of the DoubleSolenoid.

`reverseChannel` - The reverse channel of the DoubleSolenoid.

`defaultDirection` - The default direction for reloads.

### SpringableDoubleSolenoid

```
public SpringableDoubleSolenoid(int moduleNumber,
                        int forwardChannel,
                        int reverseChannel,
                        DoubleSolenoid.Value defaultDirection)
```

Initializes a new SpringableDoubleSolenoid.

**Parameters:**

`moduleNumber` - The slot number of the solenoid module.

`forwardChannel` - The forward channel of the DoubleSolenoid.

`reverseChannel` - The reverse channel of the DoubleSolenoid.

`defaultDirection` - The default direction for reloads.

## Method Detail

### getSprung

```
public boolean getSprung()
```

Has the DoubleSolenoid been sprung1

**Returns:**

Whether or not the DoubleSolenoid has been sprung.

### spring

```
public void spring ( )
```

Springs the DoubleSolenoid.

### set

```
public void set( DoubleSolenoid.Value direction)
```

Sets the direction of the DoubleSolenoid.

**Overrides:**

`set` in class `DoubleSolenoid`

**Parameters:**

`direction` ! The direction to set.

### reload

```
public void reload()
```

If the DoubleSolenoid has been sprung, unspring it; if not, set the output to the defaultoutput.

com._604robotics.utils

# Class VelocityController

java.lang.Object
    com._604robotics.utils.VelocityController

---

```
public class VelocityController
extends Object
```

Class for controlling a motor's velocity, rather than its power directly. Uses a PID loop to scale to said velocity, and a distanceE calibrated encoder for feedback.

**Author:**

Michael Smith , z evin Parker

## Constructor Summary

**Constructors**

| Constructor and Description |
| --- |
| **VelocityController**(double p, double i, double d, **Encoder** encoderLeft, **Encoder** encoderRight, **RobotDrive** robotDrive, **Gyro** gyro)<br>InitialiWes a newVelocityController. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
| --- | --- |
| void | **disable**()<br>Disables the VelocityController. |
| void | **enable**()<br>( nables theVelocityController. |
| double | **getActualVelocity**()<br>) ets the actual, current velocity. |
| double | **getVelocity**()<br>) ets the current target velocity. |
| boolean | **isEnabled**()<br>Is the VelocityController currently enabled? |
| void | **setAngleGains**(double pAngle, double iAngle, double dAngle)<br>2 ased on gyro angles TODO E javadoc |
| void | **setGains**(double p, double i, double d)<br>! econfigures the gains on the PIDController. |
| void | **setVelocity**(double velocity)<br>Sets the target velocity. |

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### VelocityController

```
public VelocityController(double p,
                  double i,
                  double d,
                  Encoder encoderLeft,
                  Encoder encoderRight,
                  RobotDrive robotDrive,
                  Gyro gyro)
```

Initializes a new VelocityController.

**Parameters:**

**Parameters:**

`p` E The proportional termfor the PIDController.

`i` E The integral termfor the PIDController.

`d` E The derivative termfor the PIDController.

`encoder` E The encoder to usefor feedback.

`output` E The PIDOutput to controlUsually some sort of motor.

## Method Detail

### getVelocity

```
public double getVelocity()
```

) ets the current target velocity.

**Returns:**

The current target velocity.

### getActualVelocity

```
public double getActualVelocity()
```

) ets the actual, current velocity.

**Returns:**

The actual, current velocity.

### setVelocity

```
public void setVelocity(double velocity)
```

Sets the target velocity.

**Parameters:**

`velocity` E The target velocity to set.

### setGains

```
public void setGains(double p,
            double i,
            double d)
```

! econfigures the gains on the PIDController.

**Parameters:**

`p` E The proportional termfor the PIDController.

`i` E The integral termfor the PIDController.

`d` E The derivative termfor the PIDController.

### setAngleGains

```
public void setAngleGains(double pAngle,
                 double iAngle,
                 double dAngle)
```

2 ased on gyro angles TODO E javadoc

**Parameters:**

`p` E The

`i` E The

`d` E The

### enable

```
public void enable()
```

( nables the VelocityController.

### disable

```
public void disable()
```

Disables the VelocityController.

### isEnabled

```
public boolean isEnabled()
```

Is the VelocityController currently enabled?

**Returns:**

- hether or not the VelocityController is currently enabled.

com._604robotics.utils

# Interface XboxController.Button.DPad

**Enclosing interface:**

XboxController.Button

---

```
public static interface XboxController.Button.DPad
```

## Field Summary

Fields

| Modifier and Type | Field and Description |
|---|---|
| static int | Down |
| static int | Left |
| static int | Right |
| static int | Up |

## Field Detail

### Up

```
static final int Up
```

**See Also:**

Constant Field Values

### Down

```
static final int Down
```

**See Also:**

Constant Field Values

### Left

```
static final int Left
```

**See Also:**

Constant Field Values

### Right

```
static final int Right
```

**See Also:**

Constant Field Values

com._604robotics.utils

# Class Gyro360

java.lang.Object
  edu.wpi.first.wpilibj.SensorBase
    edu.wpi.first.wpilibj.Gyro
      com._604robotics.utils.Gyro360

**All Implemented Interfaces:**

IDevice, ISensor, PIDSource

---

```
public class Gyro360
extends Gyro
implements PIDSource
```

Extender class to constrain the output of a Gyro to 360 degrees, looping.

**Author:**

Michael Smith

## Field Summary

### Fields inherited from class edu.wpi.first.wpilibj.SensorBase

kAnalogChannels, kAnalogModules, kDigitalChannels, kPwmChannels, kRelayChannels, kSolenoidChannels, kSolenoidModules, kSystemClockTicksPerMicrosecond

## Constructor Summary

**Constructors**

| Constructor and Description |
|---|
| **Gyro360**(AnalogChannel channel)<br>Initializes a new Gyro360 on the specified AnalogChannel. |
| **Gyro360**(int port)<br>Initializes a new Gyro360 on the specified PWM port. |
| **Gyro360**(int slot, int port)<br>Initializes a new Gyro360 on the specified PWM port on the specified module port. |

## Method Summary

**Methods**

| Modifier and Type | Method and Description |
|---|---|
| double | **getAngle**()<br>Gets the angle of the gyro, constrained to 360 degrees. |
| double | **pidGet**()<br>Implements the pidGet() function in the type PIDSource, allowing this class to be used as such. |

### Methods inherited from class edu.wpi.first.wpilibj.Gyro

free, reset, setSensitivity

### Methods inherited from class edu.wpi.first.wpilibj.SensorBase

checkAnalogChannel, checkAnalogModule, checkDigitalChannel, checkDigitalModule, checkPWMChannel, checkPWMModule, checkRelayChannel, checkRelayModule, checkSolenoidChannel, checkSolenoidModule, getDefaultAnalogModule, getDefaultDigitalModule, getDefaultSolenoidModule, setDefaultAnalogModule, setDefaultDigitalModule, setDefaultSolenoidModule

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### Gyro360

```
public Gyro360(int port)
```

Initializes a new Gyro360 on the specified PWM port. Note that port must be 1 or 2!

**Parameters:**

`port` - The PWM port the gyro is plugged into. Must be 1 or 2!

### Gyro360

```
public Gyro360(int slot,
       int port)
```

Initializes a new Gyro360 on the specified PWM port on the specified module port. Note that port must be 1 or 2!

**Parameters:**

`slot` - The module slot the gyro is plugged into.

`port` - The PWM port the gyro is plugged into. Must be 1 or 2!

### Gyro360

```
public Gyro360(AnalogChannel channel)
```

Initializes a new Gyro360 on the specified AnalogChannel. Note that port must be 1 or 2!

**Parameters:**

`channel` - The AnalogChannel the gyro is plugged into.

## Method Detail

### getAngle

```
public double getAngle()
```

Gets the angle of the gyro, constrained to 360 degrees.

**Overrides:**

`getAngle` in class `Gyro`

**Returns:**

The angle of the gyro, constrained to 360 degrees.

### pidGet

```
public double pidGet()
```

Implements the pidGet() function in the type PIDSource, allowing this class to be used as such.

**Specified by:**

`pidGet` in interface `PIDSource`

**Overrides:**

`pidGet` in class `Gyro`

**Returns:**

The angle of the gyro, constrained to 360 degrees.