Explanation of Concepts

StrangeMachine

A StrangeMachine manages the state for a particular component of the robot. They can be used in conjunction with one another to manage state changes involving multiple components, eg, the elevator and pickup positions.

A StrangeMachine has multiple "states", which represent the possible conditions the represented components can be in. You can `test` to see if a Machine is in a particular state, and `crank` a Machine toward a particular state; both operations will return whether or not the Machine is currently in the specified state.

RotationProvider

Given a PIDController that controls the turret rotation, a RotationProvider updates the setpoint of that controller, based on external feedback. This updating is done in the `update` method, which is called once per iteration in the main thread. While the PIDController is not, by default, enabled, this updating continues regardless.

Currently, the control structure works as follows:

- If the turret is in the `HIGH` position, then check if the user is pressing the `AIM_AND_FIRE` button.
- If so, make sure the PID controller is enabled and check if we're on target; if not, make sure the PID controller is disabled.
- If the `AIM_AND_FIRE` button is pressed, and we're on target, then go ahead and fire.

There are a few RotationProviders currently implemented. The one we've pretty much settled on is the `SlowbroRotationProvider`.

CameraInterface

A CameraInterface is an abstraction of a method the robot can use to obtain vision data. Currently, the only one is RemoteCameraTCP, which obtains data over a TCP connection, sent from processing software running on the Driver Station.

Springables

A Springable[Victor|Relay|DoubleSolenoid] works, in most ways, like a [Victor|Relay|DoubleSolenoid]. The only difference is the addition of a `reload` method. If the Springable[Victor|Relay|DoubleSolenoid] receives input, either through a PIDController or manually, it will put itself in the "sprung" state. When the `reload` method is called -- which it is for each Springable[Victor|Relay|DoubleSolenoid] at the end of the main control loop -- one of the following will happen:

- **if the Springable[Victor|Relay|DoubleSolenoid] is sprung:** un-spring it.
- **if the Springable[Victor|Relay|DoubleSolenoid] is not sprung:** set the output to the default output.

This way, if nothing writes to the Springable[Victor|Relay|DoubleSolenoid] over the course of a loop iteration, it will automatically switch itself off. This removes this burden from the main control logic, making things much, much simpler in implemenation.

Anything else?

Yep, there's a bunch more in here. Flip through the docs!

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)` Initializes a new GyroHax on the specified AnalogChannel.
`GyroHax(int port)` Initializes a new GyroHax on the specified PWM port.
`GyroHax(int slot, int port)` 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()` 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``()` 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)` Initializes a new PIDDriveEncoderDifference, based on the given encoders.

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`pidGet()` 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:

`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)` Initializes a new PIDDriveGyro, based on the given RobotDrive.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`pidWrite(double output)` 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)` Initializes a new PIDDriveEncoderOutput.
`PIDDriveEncoderOutput(RobotDrive driveTrain, boolean inversion)` Initializes a new PIDDriveEncoderOutput.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`pidWrite(double output)` 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)` 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)` 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)` 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)` 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)` 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:

`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)` 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)` 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)` 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)` Causes the Machine to strive for the target state.
`boolean`	`test(int state)` 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`** The possible states the shooter could be in.

Constructor Summary

Constructors

Constructor and Description
`ShooterMachine``(`**`DualVictor`**` shooter, `**`Victor`**` hopper)` Initializes a new ShooterMachine.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	**`crank`**`(int state)` Causes the Machine to strive for the target state.
`void`	**`setShooterSpeed`**`(double speed)` Sets the shooter speed to use when, well, shooting.
`boolean`	**`test`**`(int state)` 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`** The possible states the turret could be in.

Constructor Summary

Constructors

Constructor and Description
`TurretMachine``(`**`PIDController`**` controller, `**`RotationProvider`**` provider, `**`Encoder`**` encoder)` Initializes a new TurretMachine.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	**`crank`**`(int state)` Causes the Machine to strive for the target state.
`void`	**`setTurretSidewaysPosition`**`(double turretSidewaysPosition)` Sets the position to use as "SIDEWAYS".
`boolean`	**`test`**`(int state)` 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:

`ckntroller` - The PIDController to control.

`prov ider` - 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 v kidsetTurretSidewaysPksitikn(dk uble turretSidewaysPksitikn)`

Sets the position to use as "SIDEWAYS".

Parameters:

`turretSidewaysPksitikn` - 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` Possible states the pickup could be in.

Constructor Summary

Constructors

Constructor and Description
`PickupMachine(DoubleSolenoid pickup)` Initializes a new PickupMachine.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`crank(int state)` Causes the Machine to strive for the target state.
`boolean`	`test(int state)` 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`** Various possible states the elevator can be in.

Constructor Summary

Constructors

Constructor and Description
`ElevatorMachine``(`**`PIDController`**` controller, `**`Encoder`**` encoder)` Initializes a new ElevatorMachine.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	**`crank`**`(int state)` Causes the Machine to strive for the target state.
`boolean`	**`test`**`(int state)` 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(PW Dontroller 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 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 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_DRIVERSTATION_PROP, ROBOT_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, double upperBand, double lowerBand, double correctValue)` If a value is within a range, set it to a specific value.
`void`	`disabled()` The robot is disabled.
`static boolean`	`isInRange(double xValue, double upperRange, double 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

`robotMain, startCompetition`

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

`destroyApp, free, getBooleanProperty, getWatchdog, isAutonomous, isDisabled, isEnabled, isNewDataAvailable, isOperatorControl, isSystemActive, pauseApp, startApp`

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)` Initializes a new SlowbroRotationProvider.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`setDefaultPosition(double defaultPosition)` Sets the "default" position, if no targets can be located.
`boolean`	`update()` 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)` Initializes a new NaiveRotationProvider, giving it control over the specified PIDController.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`setDefaultPosition(double defaultPosition)` Sets the "default" position, if no targets can be located.
`boolean`	`update()` 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`
U pdates 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)` Initializes a new DummyRotationProvider, giving it control over the specified PIDController.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`setDefaultPosition(double defaultPosition)` Sets the "default" position, if no targets can be located.
`boolean`	`update()` 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)` Sets the "default" position, if no targets can be located.
`boolean`	`update()` 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 cameral nterface, Encoder encoderTurret)` Initializes a new SlightlySmarterRotationProvider.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`setDefaultPosition(double defaultPosition)` Sets the "default" position, if no targets can be located.
`boolean`	`update()` W pdates the aiming 6the turret.

Methods inherited from class java.lang.Object

`clone, eq als, finaliz e g eClass, hashCode, notify, notifyAll, tof ring, wait, wait, wait`

Constructor Detail

SlightlySmarterRotationProvider

```
public f lightlyf marterRotationProvider(PI DoCtroller controller,
                                         Cameral nterface cameral nterface,
                                         S noder encoderTurret)
```

Initializes a new SlightlySmarterRotationProvider.

Parameters:

`controller` - The PIDController to control.

`cameral nterface` - The CameraInterface to read data from.

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

Method Detail

setDefaultPosition

```
public void setD €aultPosition(double defaultPosition)
```

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

Specified by:

`setD €aultPosition` in interface `RotationProvider`

update

```
public boolean update()
```

Description copied from interface: `RotationProvider`
W pdates the aiming 6the 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 MEDIUMzUPPE_
```

See Also:

Constant Field Values

MEDIUM_LOWER

```
static final int MEDIUMz K A_Y
```

See Also:

Constant Field Values

LOW

```
static final int K A Y
```

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 MEDIUMzUPPE_
```

See Also:

Constant Field Values

MEDIUM_LOWER

```
static final int MEDIUMz K A_Y
```

See Also:

Constant Field Values

LOW

```
static final int K A Y
```

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`

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`ELEVATOR_A`
`static int`	`ELEVATOR_B`
`static int`	`TURRET_ROTATION_A`
`static int`	`TURRET_ROTATION_B`

Field Detail

ELEM p TO F _ p

```
static final int El H o M , y z ) M
```

See p Iso

Constant Field Values

ELEM p TO F _ B

```
static final int El H o M , y z ) v
```

See p Iso

Constant Field Values

TURRETvRBTATIBNvA

```
static final int , q z z H , ) z y , M , f y g ) M
```

See Also:

Constant Field Values

TURRETvRBTATIBNv U

```
static final int , q z z H , ) z y , M , f y g ) v
```

See Also:

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 Dl o M ,
```

See Also:

Constant Field Values

MANIPULATOR

```
static final int y z ) o v q f z g K l
```

See Also:

Constant Field Values

com._604robotics.robot2012.configuration

Interface ActuatorConfiguration.SOLENOID_SHIFTEm

Enclosing interfacey

ActuatorConfiguration

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

Field Summary

Fields

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

Field Detail

LOW_GEAm

```
static final DoubleSolenoid.Value LOW_ K A Y _
```

HIGH_GEAm

```
static final DoubleSolenoid.Value H, G) _ K A Y _
```

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 H I uble	BACKWARD_DISTANCE
static H I uble	BACKWARD_DISTANCE_SIDES
static H I uble	BACKWARD_DRIVE_POWER
static H I uble	FORWARD_DISTANCE
static H I uble	FORWARD_DRIVE_POWER

Field Detail

FORWARD_DISTANCE

```
static final H I uble o M , y z , ) v ) q f g z K h A
```

See Also:

Constant Field Values

BACKWARD_DISTANCE

```
static final H I uble R z h N y z , ) v ) q f g z K h A
```

See Also:

Constant Field Values

BACKWARD_DISTANCE_SIDES

```
static final H I uble R z h N y z , ) v ) q f g z K h A v f q ) A f
```

See Also:

Constant Field Values

FORWARD_DRIVE_POWER

```
static final H I uble o M , y z , ) v ) , q w A v R M y A ,
```

See Also:

Constant Field Values

BACKWARD_DRIVE_POWER

```
static final H I uble R z h N y z , ) v ) , q w A v R M 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 H l uble`	`ACCELEROMETER_SENSITIVITY`
`static H l uble`	`ACCELEROMETER_UPPER_RADIANS`
`static H l uble`	`GYRO_DRIFT`
`static int`	`TURRET_CALIBRATION_OFFSET`

Field Detail

GYRO_DRIFT

```
static final H l uble o M , y z ) , v q f
```

See Also:

Constant Field Values

ACCELEROMETER_SENSITIVITY

```
static final H l uble g K K h A h , y R h f h , z N h w N v f v R v f M
```

See Also:

Constant Field Values

ACCELEROMETER_UPPER_RADIANS

```
static final H l uble g K K 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 K 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_DRIBE

```
static final int LEFT_Dz ) v I
```

See Also:

Constant Field Values

RIGHT_DRIBE

```
static final int z ) q f M , y z ) v I
```

See Also:

Constant Field Values

ELEB MUR_LEFT

```
static final int ELEg ADz , H I o M
```

See Also:

Constant Field Values

ELEB MUR_RIGHT

```
static final int ELEg ADz , z ) q f M
```

See Also:

Constant Field Values

SP U ERT_LEFT

```
static final int y HD DzTE H I o M
```

See Also:

Constant Field Values

SP U ERT_RIGHT

```
static final int y HD DzTE z ) q f M
```

See Also:

Constant Field Values

P U ER

```
static final int HD ) z E
```

See Also:

Constant Field Values

PICh U P

```
static final int )z L E U T
```

See Also:

Constant Field Values

TURRET_RU ATIU N

```
static final int TSz z I M , z K M g M ) K 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` H I o M , o y

See Also:

Constant Field Values

LEFT

`static final int` LzH v

See Also:

Constant Field Values

RIGHT

`static final int` R_ K A g

See Also:

Constant Field Values

DOWN

`static final int` DOM K

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 H I o M , 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 TM , , y o z ) y v , y y q z f y , z g K h g A
```

See Also:

Constant Field Values

LEFT_DRIVE_INCHES_PER_CLICK

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

See Also:

Constant Field Values

RIGHT_DRIVE_INCHES_PER_CLICK

```
static final double , h v R o z ) , h N y z h w g R y q z f y , z g K 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`

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`AIM_AND_SHOOT`
`static int`	`PICKUP`
`static int`	`TOGGLE_ANGLE`
`static int`	`TOGGLE_HEIGHT`
`static int`	`TOGGLE_LIGHT`

Field Detail

M M_M NmSβ MTM

```
static final int H I o M H , y M z ) v v q
```

See M lso

Constant Field Values

PICv U P

```
static final int PIH DPy
```

See M lso

Constant Field Values

TM U E_BEIGBT

```
static final int _ g T M) N I A ) q
```

See M lso

Constant Field Values

TM U E_M NGEL

```
static final int _ g T M H E A R N
```

See M lso

Constant Field Values

TM U E_L IBT

```
static final int _ g T TMRE E A ) q
```

See M lso

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`

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`COMPRESSOR`
`static int`	`PRESSURE_SWITCH`

Field Detail

COMPRESSOR

```
static final int H I o M , y z z I ,
```

See Also:

Constant Field Values

PRESSUREMSWITCP

```
static final int P, y z z ) , y v z q f g H K
```

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	REGYLAR

Field Detail

REGULAR

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

PUSH

```
static final DoubleSolenoid.Value P_Sy
```

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 g _ K
```

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 FM , y z , )`

See Also:

Constant Field Values

LEFT

`static final double g FK`

See Also:

Constant Field Values

RIGHT

`static final double , g K h f`

See Also:

Constant Field Values

TOLERANCE

`static final double KM 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 H l uble	`ACCELEROMETER_DRIVE_POWER`
static H l uble	`ELEVATOR_POWER_MAX`
static H l uble	`ELEVATOR_POWER_MIN`
static H l uble	`HOPPER_POWER`
static H l uble	`HOPPER_POWER_REVERSE`
static H l uble	`PICKUP_POWER`
static H l uble	`TURRET_ROTATION_POWER_MAX`
static H l uble	`TURRET_ROTATION_POWER_MIN`

Field Detail

ACCELEROMETER_DRIVE_POWER

static final H l uble o M M , y , z) v , q , z f g z K h , f A) R , z

See Also:

Constant Field Values

HOPPER_POWER

static final H l uble N) A A , z f A) R , z

See Also:

Constant Field Values

HOPPER_POWER_REVERSE

static final H l uble N) A A , z f A) R , z f z , h , z w ,

See Also:

Constant Field Values

PICKUP_POWER

`static final H I uble A K M R U A f A ) R , z`

See Also:

Constant Field Values

ELEVATOR_POWER_MIN

`static final H I uble , y , h o q ) z f A ) R , z f v K N`

See Also:

Constant Field Values

ELEVATOR_POWER_MAX

`static final H I 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 H I uble q U z z , q f z ) q o q K ) N f A ) R , z f v K N`

See Also:

Constant Field Values

TURRET_ROTATION_POWER_MAX

`static final H I uble q U z z , q f z ) q o q K ) 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

GYRO_HEP DIKG

```
static final int GI o M , y z ) v q f H
```

See P Iso

Constant Field Values

GYRO_BP W P ENC

```
static final int GI o M , g ) K ) f h z
```

See P Iso

Constant Field Values

P CCEWERBMETER

```
static final int z _z_K z o M A z R z o
```

See P Iso

Constant Field Values

EWEK PBTRM LMITMSY ITCH

```
static final int z K z N ) R M o , K q A q R , w R q R h y
```

See P 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` H I

See v le:

Constant Field Values

OUT

`static final int` OM ,

See v le:

Constant Field Values

com._604robotics.robot2012.configuration

Interface ButtonConfiguration.Driver

Enclosing interface:

ButtonConfiguration

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

Field Summary

Fields

Modifier and Tyy e	Field and Descriy iton
`static int`	`AUTO_BALANCE`
`static int`	`GYRO_RESET`
`static int`	`SHIFT`
`static int`	`TOGGLE_PICKUP`

Field Detail

SHIFT

```
static final int SHo M ,
```

See Also:

Constant Field Values

TB M EPLPICU W P

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

See Also:

Constant Field Values

AWTB BALANCE

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

See Also:

Constant Field Values

M Y R ESETR

```
static final int GS y q R v H 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 LOo M , y z )
```

See Also:

Constant Field Values

HIGH_GEAR

```
static final int g , v M , 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** the x value
double	**y** the y value
double	**z** 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**(z
double	**getY**(z
double	**getZ**(z
goid	**setX**(double xz Sets the X value of this Point
goid	**setY**(double y) Sets the Y value of this Point
v odi	**setZ**(double z) Sets the Z value of this Point

Methods inherited from class java.lang.Object

clOne, eq als, f inaliz e g eD ass, hasy Dde, notify, notify A lltoStA iH, 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 G G the x value

y G G the y value

z G G the z value

Method Detail

getX

```
public double getX()
```

Returns:

G theX 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 getZ()
```

Returns:

G the(value

setZ

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

Sets the (value of this Point

Parameters:

`z` - - the Z 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** This is the angle of the target, relative to the camera.
double	**angle_uncertainty** This is the uncertainty of the angle of the target.
double	**x** x, y, and z represent the W (pdosition of the target x will be positive when the target appears to be right of the center of the camera.
double	**x_uncertainty** These are the uncertainties of the x, y, and z positions of the target.
double	**y** x, y, and z represent the W (pdosition of the target x will be positive when the target appears to be right of the center of the camera.
double	**y_uncertainty** These are the uncertainties of the x, y, and z positions of the target.
double	**z** x, y, and z represent the W (pdosition of the target x will be positive when the target appears to be right of the center of the camera.
double	**z_uncertainty** These are the uncertainties of the x, y, and 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.

Gangle1
......GTarget1
.......2
......2
.....2
....2-------|! GCamera1
...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 (pdosition 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 G 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 - xy, z> _accuracyThe units of these measures are in inches.

y

```
public double y
```

x, y, and z represent the W (pdosition 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 G 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 - xy, z> _accuracyThe units of these measures are in inches.

z

```
public double z
```

x, y, and z represent the W (pdosition 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 G 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 - xy, z> _accuracyThe 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 Sarget(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``()` Initiali- es a new Point3d.
`Point3d``(double x, double y, double z)` Initiali- es a new Point3d.

Method Summary

Methods inherited from class java.lang.Object

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

Field Detail

x

```
public double x
```

y

```
public double y
```

z

```
public double z
```

Constructor Detail

Point3d

```
public Sointr d()
```

Initiali- es a new Point3d.

Point3d

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

Initiali- es a new Point3d.

Parameters:

`x` K The xK coordinate oz the point.

`y` K The yK coordinate oz the point.

`z` K The - K coordinate oz 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)` 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 x- coordinate oK the point.

`y` - The y- coordinate oK 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)` Initializes variables for the point.
`PointAndAngle3d(Point3d p, double angle)` Initiali- esvariables 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 PointAndAngle3d(double x,
                       double y,
                       double z,
                       double angle)
```

Initiali- esvariables for the point.

Parameters:

`x` WThe x coordinate of the point.

`y` WThe y coordinate of the point.

`z` WThe - coordinate of the point.

`angle` WThe angle the target is at from the robot.

PointAndAngle3d

```
public PointAndAngle3d(Point3d p,
                       double angle)
```

Initiali- esvariables for the point.

Parameters:

`p` W(sesthe values from this point to create the new point.

`angle` W(sesthis 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)` - ethe angle from the targetsKand 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)` This function gets the direction the target is facingKrelative to the camera.
`Point3d`	`getRelXYZOfTarget(double x1, double y1, double w, double h)` W emembethat this re(uiresthe camera to be G pefectlyGflatKand the targets to be G pefectlyG 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

```
public Aiming()
```

Method Detail

getRelXYZOfTarget

```
public Point3d getRelXYZOfTarget(double x1,
                                 double y1,
                                 double w,
                                 double h)
```

W emembethat this re(uiresthe camera to be G pefectlyGflatKand the targets to be G pefectlyG vertical. A newfunction will probably need to be created for use on the robot. ThatKor we1 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- valuef the bottom left corner

`y1` - y- valuef the bottom left corner

`w` - width fthe vision target

`h` - height fthe vision target

Returns:

a Point- d holding the XYKand Z fthe targetKrelative 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 facingKrelative to the camera. It is imperfectKand half- assumes a simple orthographic prjection (which is not (uitelike real life) .flit causes issues (which the accuracy fthis function doesn1 t need to be very highKwe can fix it later.

Parameters:

`x1` - x- valuef the bottom left corner

`y1` - y- valuef 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)
```

- ethe angle from the targetsKand 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()` Launches the CameraInterface.
`void`	`end()` Disables the CameraInterface.
`double`	`getRecordedTime()` Gets the estimated time since the last packet was received.
`Target[]`	`getTargets()` 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()` Ends communication.
`double`	`getRecordedTime()` Records the time elapsed between reception of data packets from camera.
`Target[]`	`getTargets()` Returns the last targets acq uiredfrom the remote software.
`int`	`getUPS()` 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:

`begin` in interface `CameraInterface`

end

```
public void end()
```

Ends communication.

Specified by:

`end` in interface `CameraInterface`

getTargets

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

Returns the last targets acq uiredfrom the remote software.

Specified by:

`getTargets` in interface `CameraInterface`

Returns:

The last targets acq 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)` Initializes a new SpringableRelay.
`SpringableRelay(int moduleNumber, int channel, Relay.Value defaultDirection)` Initializes a new SpringableRelay.
`SpringableRelay(int channel, Relay.Direction direction, Relay.Value defaultDirection)` Initializes a new SpringableRelay.
`SpringableRelay(int channel, Relay.Value defaultDirection)` Initializes a new SpringableRelay.

Method Summary

Methods

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

void spring()
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:

`channel` - The channel the Relay is on.

`devaultDirection` - The default direction for reloading.

v etod Detail

getSprung

```
public boolean getSprung()
```

Has the Relay been sprung?

Returns:

- hether or not the Relay 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 itR 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)` Initiali- esa new CompensatingGyro on the specified AnalogChannel.
`CompensatingGyro(int port)` Initiali- esa new CompensatingGyro on the specified PWM port.
`CompensatingGyro(int slot, int port)` Initiali- esa new CompensatingGyro on the specified PWM port on the specified module port.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`setAccumulatorCenter(int center)` 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 jav alang.Object

`clone, eT uals, f inaliz,egetClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

CompensatingGyro

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

Initiali- esa new CompensatingGyro on the specified PWM port. Note that port must be Wor ( )

Parameters:

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

CompensatingGyro

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

Initiali- esa new CompensatingGyro on the specified PWM port on the specified module port. Note that port must be Wor ( )

Parameters:

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

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

CompensatingGyro

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

Initiali- 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)` Initializes a new DeadbandedSource.

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`pidGet()` Hooks into PIDSource - gets the value to send to the PIDController.
`void`	`setController(PIDController controller)` Sets the PIDController the source is fed into.
`void`	`setDeadband(double lowerDeadband, double upperDeadband)` Sets the range for the deadband.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, wasw ɒde, notifS, notifS l ,lltoStrinS, wait, wait, wait`

Constructor Detail

DeadbandedSource

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

Initializes a new DeadbandedSource.

Parameters:

`source` - The underlying PIDSource to wrap around.

Method Detail

setController

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

Sets the PIDController the source is fed into.

Parameters:

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

setDeadband

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

Sets the range for the deadband.

Parameters:

`lowerDeadband` - The lower bound of the deadband.

`upperDeadband` - The upper bound of the deadband.

pidGet

```
public double pid( et()
```

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

Specified by:

`pid( et` in interface `PIDSource`

Returns:

The value to send to the PIDController.

Overview Package Class Tree Deprecated Index Help

Prev Class **Next Class** Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

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` 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)` Initializes a new UpDownPIDController.

Method Summary

Methods

Modifier and Type	Method and Description
`UpDownPIDController.Gains`	`getDownGains()` Gets the Gains for going down.
`UpDownPIDController.Gains`	`getUpGains()` Gets the Gains for going up.
`void`	`refreshGains()` Updates the gains for the current direction.
`void`	`setDownGains(UpDownPIDController.Gains downGains)` Sets the gains for going down.
`void`	`setSetpoint(double setpoint)` Sets the setpoint to go to.
`void`	`setUpGains(UpDownPIDController.Gains upGains)` 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` - The gains to use when going up.

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

`source` - The PIDSource to plug in.

`output` - 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` - The gains to use when going up.

setDownGains

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

Sets the gains for going down.

Parameters:

`downGains` - 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` - 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 - see SpringableVictorK .

Constructor Summary

Constructors

Constructor and Description
`DualVictor(int leftPort, int righ Ꮀort)` Initializ e a DualVictor with a left and a right PWM port.
`DualVictor(int leftS ꝺt, int leftPort, int righ tSoꝉ, int righ Ꮀort)` InitialiWes a DualVictor with left and right slot and PWM port.
`DualVictor(Victor leftVictor, Victor righ tV icr)` InitialiWes a DualVictor with left and right slot and PWM port.

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`get()` Checks the current power the Victors are set to.
`boolean`	`getSprung()` Has the victor been sprungG
`Aoid`	`pidWrite(double output)` Function to hook into the PIDController.
`Aoid`	`reload()` If the Victor has been sprung, unspring it1if not, set the output to 0.
`Aoid`	`set(double speed)` Sets the power of the Victors.
`Aoid`	`setController(PIDController controller)` Sets the PIDController for this DualVictor, if there is one.
`Aoid`	`setDeadband(double loSerDeadband, double upperDeadband)` Sets the deadband for the DualVictor.
`Aoid`	`setLeftInversion(boolean inAersion)` Sets the inversion for the 2 left2Victor.
`Aoid`	`setRightInversion(boolean inAersion)` Sets the inversion for the 2 right2Victor.
`Aoid`	`setSafetyEnabled(boolean enabled)` Sets whether or not safety is enabled.
`Aoid`	`spring()` Springs the victor.

Methods inherited from class java.lang.Object

`clone, er als, finaliWe, getS ꝉass, hash Code, notifl , notifl ( ,lltoString, Sait, Sait, Sait`

Constructor Detail

DualVictor

```
public DualVictor(int leftPort,
          int righ tPort)
```

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

Parameters:

`leftPort` ! The PWM port of the 2 left2Victor.

`righ tPort` ! The PWM port of the 2 right2Victor.

DualVictor

```
public DualVictor(int leftS lot,
          int leftPort,
          int righ tSlot,
          int righ tPort)
```

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

Parameters:

`leftS lot` ! The slot the 2 left2Victor is plugged into.

`leftPort` ! The PWM port of the 2 left2Victor.

`righ tSlot` ! The slot the 2 right2Victor is plugged into.

`righ tPort` ! The PWM port of the 2 right2Victor.

DualVictor

```
public DualVictor(Victor leftVictor,
          Victor righ tV ictor)
```

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

Parameters:

`leftVictor` ! The 2 left2Victor.

`righ tV ictor` ! The 2 right2Victor.

Method Detail

getSprung

```
public boolean getS prung()
```

Has the victor been sprungG

Returns:

Whether or not the victor has been sprung.

spring

```
public Aoid spring()
```

Springs the victor.

setLeftInversion

```
public Aoid setL eftInAersion(boolean inAersion)
```

Sets the inversion for the 2 left2Victor.

Parameters:

`inAersion` ! Is it invertedG

setRightInversion

```
public Aoid setR igD tInAersion(boolean inAersion)
```

Sets the inversion for the 2 right2Victor.

Parameters:

`inAersion` ! Is it invertedG

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:

`pidf rite` in interface `PIDOutput`

Parameters:

`output` ! The speed to set.

setDeadband

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

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

Parameters:

`loSerDeadband` ! The lower bound of the deadband.

`upperDeadband` ! The upper bound of the deadband.

setSafetyEnabled

```
public Aoid setSafetl Eabled(boolean enabled)
```

Sets whether or not safety is enabled.

Parameters:

`enabled` ! Whether or not safety is enabled.

reload

```
public Aoid reload()
```

If the Victor has been sprung, unspring it1if not, set the output to 0.

setController

```
public Aoid setC ontroller(PIDSontroller 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 PIDController for 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
```

Extender of a Victor providing an easier control flow. When an output is set for the Victor, it is considered K sprungK When the K reloadK method is called if the victor is sprung, it unsprings the Victor. If the Victor is not sprung, then the output is set to Wero. In this way the Victor will only be moving when you tell it to. (se this in a loop or something, and call K reloadK at the end. No more worries about code pat that donGt update the Victors1

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)` InitialiWes a new SpringableVictor on the given PWM port.
`SpringableVictor``(int slot, int port)` InitialiWes a new SpringableVictor on the given module slot and PWM port.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`getSprung()`

boolean	`getSprung()` Has the victor been sprung2
void	`pidWrite(double output)` Function to hook into the PIDController.
void	`reload()` If the Victor has been sprung, unspring it! if not, set the output to 0.
void	`set(double speed)` Sets the power of the Victor.
void	`setController(PIDController controller)` Sets the PIDController for this Victor, if there is one.
void	`spring()` Springs the victor.

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

`get, set`

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

`disable, Veed, YetDescription, 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)
```

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

Parameters:

`slot` - The module slot the Victor is connected to.

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

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.

set

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

Sets the power of the Victor.

Specified by:

`set` in interface `SpeedController`

Overrides:

`set` in class `Victor`

Parameters:

`speed` - The speed to set.

pidWrite

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

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

Specified by:

`pidWrite` in interface `PIDOutput`

Overrides:

`pidWrite` in class `Victor`

Parameters:

`output` - The speed to set.

reload

```
public void reload()
```

If the Victor has been sprung, unspring it! if not, set 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 itGs updating it, and wonGt 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)` 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)` Allocate a PID object with the given constants for P, I, D

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`getRealSetpoint()` Gets the "real" setpoint of the PIDController.
`double`	`getSetpoint()` Returns the current setpoint of the PIDController
`void`	`setConversionFactor(double conversionFactor)` Sets the factor to use when doing conversion on setSetpoint and getSetpoint.
`void`	`setRealSetpoint(double setpoint)` Sets the "real" setpoint of the PIDController.
`void`	`setSetpoint(double setpoint)` 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 ConDertingPIDController(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: `eH uZpd.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.

Parameters:

`setpoint` W The "real" setpoint to set.

setSetpoint

```
public Doid setvetpoint(double setpoint)
```

Description copied from class: `eH uzpd.first.wpilibj.PIDController`
Set the setpoint for the PIDController

Overrides:

`setS etpint` in class `PIDController`

Parameters:

`setpoint` W the desired setpoint

setConversionFactor

```
public Doid setConDersionyactor(double conDersionyactor)
```

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

Parameters:

`conDersionyactor` W The conversiorfactor 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. Else, if weKre within a certain z coasting range, the output will be floored at the Wcoasting outputW. Else, if weKre outside a certain z hozintal rangeW, the output will be ceilinged at a certain Whozontal outputW. Else, 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)` Initializes a new LinearController.

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`calculate()` Function that performs the output calculation.
`double`	`getTarget()` (ets the current target.
`boolean`	`onTarget()` Are we there yet)
`Hoid`	`SetCoaS tngRange(double coastingRange, double coastingOutput)` 1 pdates the coasting values.
`Hoid`	`SetH riz ntalRange(double horizontalRange, double horizontalOutput)` 1 pdates the hozontal values.
`Hoid`	`SetTarget(double target)` Sets the current target.
`Hoid`	`update()` 1 pdates the PIDOutput based on the latest data.

Methods inherited from class java.lang.Object

`clone, equals, y ialize, getClass, hashCode, notiy , notiy , A, ltoString, r ait, r ait, r ait`

Constructor Detail

LinearController

```
public W nearController(PI ( oSrce source,
                        PI (Output output,
                        double horizontalRange,
                        double horizontalOutput,
                        double coastingRange,
                        double coastingOutput)
```

Initializes a new LinearController.

Parameters:

`source` 2A PIDSource to read from.

`output` 2A PIDOutput to write to.

`horizontalRange` 2The horizontal range, as defined in the class description.

`horizontalOutput` 2The horizontal output, as defined in the class description.

`coastingRange` 2The coasting range, as defined in the class description.

`coastingOutput` 2The coasting output, as defined in the class description.

Method Detail

setBorizontalRange

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

1 pdatesthe horizontal values.

Parameters:

`horizontalRange` 2The horizontal range, as defined in the class description.

`horizontalOutput` 2The horizontal output, as defined in the class description.

setCoastingRange

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

1 pdatesthe coasting values.

Parameters:

`coastingRange` 2The coasting range, as defined in the class description.

`coastingOutput` 2The coasting output, as defined in the class description.

getTarget

```
public double getT arget()
```

(ets the current target.

Returns:

The current target.

setTarget

```
public Hoid setvarget(double target)
```

Sets the current target.

Parameters:

`target` 2 The target to move toward.

onTarget

```
public boolean onvarget()
```

Are we there yet)

Returns:

Whether or not weKre 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 Hoid update()
```

1 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)` Encoder constructor.
`EncoderPIDSource(DigitalSource aSource, DigitalSource bSource, boolean reverseDirection)` Encoder constructor.
`EncoderPIDSource(DigitalSource aSource, DigitalSource bSource, boolean reverseDirection, CounterBase.EncodingType encodingType)` Encoder constructor.
`EncoderPIDSource(DigitalSource aSource, DigitalSource bSource, DigitalSource indexSource)` Encoder constructor.
`EncoderPIDSource(DigitalSource aSource, DigitalSource bSource, DigitalSource indexSource, boolean reverseDirection)` Encoder constructor.
`EncoderPIDSource(int aChannel, int bChannel)` 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()` 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` 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 6the 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(,)kX G , or k4G to indicate R G , X G or 4G decoding. If 4G is selected, then an encoder FPK A object is used and the returned counts will be 4x the encoder spec2 d value since all rising and falling edges are counted. If R G or X G are 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` 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(,)kX G , or k4G to indicate R G , X G or 4G decoding. If 4G is selected, then an encoder FPK A object is used and the returned counts will be 4x the encoder spec2 d value since all rising and falling edges are counted. If R G or X G are 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. ! singan 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.

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 usedfor the a channel.

`bSource` W the source that should be usedfor the b channel.

`reverseDirection` W represents the orientation 6the 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 usedfor the a channel.

`bSource` W the source that should be usedfor 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 usedfor the a channel.

`bSource` W the source that should be usedfor the b channel.

`reverseDirection` W represents the orientation 6the encoder and inverts the output values if necessary so forward represents positive values.

`encodingType` W either k(,)kX G , or k4G to indicate R G , X G or 4G de4Godingsselected, then an encoder FPK A object is usedand the returned counts will be 4x the encoder spec2 d value since allrising and falling edges are counted. If R G or X G are 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 usedfor the a channel.

`bSource` W the source that should be usedfor the b channel.

`indexSource` W the source that should be usedfor 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)` Encoder constructor.
`EncoderOffset(DigitalSource aSource, DigitalSource bSource, boolean reverseDirection)` Encoder constructor.
`EncoderOffset(DigitalSource aSource, DigitalSource bSource, boolean reverseDirection, CounterBase.EncodingType encodingType)` Encoder constructor.
`EncoderOffset(DigitalSource aSource, DigitalSource bSource, DigitalSource indexSource)` Encoder constructor.
`EncoderOffset(DigitalSource aSource, DigitalSource bSource, DigitalSource indexSource, boolean reverseDirection)` Encoder constructor.
`EncoderOffset(int aChannel, int bChannel)`

Encoder constructor.
EncoderOffset(int aChannel, int bChannel, boolean reverseDirection) Encoder constructor.
EncoderOffset(int aChannel, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType) Encoder constructor.
EncoderOffset(int aChannel, int bChannel, int indexChannel) Encoder constructor.
EncoderOffset(int aChannel, int bChannel, int indexChannel, boolean reverseDirection) Encoder constructor.
EncoderOffset(int aSlot, int aChannel, int bSlot, int bChannel) Encoder constructor.
EncoderOffset(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection) Encoder constructor.
EncoderOffset(int aSlot, int aChannel, int bSlot, int bChannel, boolean reverseDirection, **CounterBase.EncodingType** encodingType) Encoder constructor.
EncoderOffset(int aSlot, int aChannel, int bSlot, int bChannel, int indexSlot, int indexChannel) Encoder constructor.
EncoderOffset(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
int	**getRaw**() K ets the raw value from the encoder.
void	**reset**() R esets the Encoder.
void	**setOffset**(int offset) 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) , k2 X , or k4X to indicate G X , 2 X or 4X decoding. If 4X is selected, then an encoder FPK A object is used and the returned counts will be 4x the encoder specRd value since all rising and falling edges are counted. If G X or 2 X are selected then a counter object will be used and the returned value will either exactly match the specRd count or be double (2 x the specRd 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) , k2 X , or k4X to indicate G X , 2 X or 4X decoding. If 4X is selected, then an encoder FPK A object is used and the returned counts will be 4x the encoder specRd value since all rising and falling edges are counted. If G X or 2 X are selected then a counter object will be used and the returned value will either exactly match the specRd count or be double (2 x the specRd 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

```
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 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.

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) , k2 X , or k4X to indicate G X , 2 X or 4X decoding. If 4X is selected, then an encoder FPK A object is used and the returned counts will be 4x the encoder specRd value since all rising and falling edges are counted. If G X or 2 X are selected then a counter object will be used and the returned value will either exactly match the specRd count or be double (2 x the specRd 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
K ets the raw value from the encoder. The raw value is the actual count unscaled by the G x, 2 x, or 4x scale factor.

Overrides:

getRaw in class Encoder

Returns:

Current raw count from the encoder

reset

```
public void reset()
```

R 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, DetClass, hashCode, notify, notifyAll, toStrinD, wait, wait, wait`

Field Detail

P

```
public double P
```

I

```
public double I
```

D

```
public double D
```

Constructor Detail

UpDownPIDController.Gains

```
public UpDownPIDController(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` Enumeration for the available axes on the Xbox controller.
`static interface`	`XboxController.Button` Enumeration for the available buttons on the Xbox controller.
`static interface`	`XboxController.Stick` Enumeration for the available sticks on the Xbox controller.

Constructor Summary

Constructors

Constructor and Description
`XboxController(int port)` Initialize a new XboxController on the specified port.
`XboxController(Joystick joystick)` Initialize a new XboxController from the underlying Joystick.

Method Summary

Methods

Modifier and Type	Method and Description
`double`	`getAxis(int axis)` Get the value of the specified axis.
`boolean`	`getButton(int button)` Get whether or not the specified button is currently pressed.
`Joystick`	`getJoystick()` Gets the underlying Joystick object.
`boolean`	`getStick(int stick)` Get whether or not there's a value reading on the stick.
`boolean`	`getToggle(int button)` Get the toggle state of the specified button.
`void`	`resetToggles()` Resets the toggle registry for the contrller.
`void`	`setDeadband(int axis, double lower, double upper)` 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.

getSticB

`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.

getJoysticB

```
public Joystick getJoystick()
```

W ets the underlyingJoystick object. What, is XboxController not good enough for you?

M **etrns:**

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 RIGH TSTICK`

See Also:

Constant Field Values

DPAD

`static final int DP) D`

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 Also:

Constant Field Values

B

```
static final int B
```

See Also:

Constant Field Values

X

```
static final int X
```

See Also:

Constant Field Values

Y

```
static final int Y
```

See Also:

Constant Field Values

LB

```
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

LeftStick

```
static final int LeftStick
```

See Also:

Constant Field Values

RightStick

```
static final int RightStick
```

See Also:

Constant Field Values

LT

```
static final int LT
```

See Also:

Constant Field Values

RT

```
static final int RT
```

See Also:

Constant Field Values

EitherTrigger

```
static final int EitherTrigger
```

See Also:

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.SolenoidB ase

`m_allocated, m_moduleNumber`

Fields inherited from class edu.wpi.first.wpilibj.SensorB ase

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

Constructor Summary

Constructors

Constructor and Description
`SpringableDoubleSolenoid(int forwardChannel, int reverseChannel, DoubleSolenoid.Value defaultDirection)` Initializes a new SpringableDoubleSolenoid.
`SpringableDoubleSolenoid(int moduleNumber, int forwardChannel, int reverseChannel, DoubleSolenoid.Value defaultDirection)` Initializes a new SpringableDoubleSolenoid.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`getSprung()` Has the DoubleSolenoid been sprung?
`void`	`reload()` If the DoubleSolenoid has been sprung, unspring it; if not, set the output to the default output.

void	**set**(**DoubleSolenoid.Value** direction) Sets the direction of the DoubleSolenoid.
void	**spring**() 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)
```

InitialiGes 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)
```

InitialiGes 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 default output.

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 distance- calibrated encodefor feedback.

Author:

Michael Smith , K evin Parker

Constructor Summary

Constructors

Constructor and Description
`VelocityController(double p, double i, double d, Encoder encoderLeft, Encoder encoderRight, RobotDrive robotDrive, Gyro gyro)` Initializ es a newVelocityController.

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`disable()` Disables the VelocityController.
`void`	`enable()` (nables theVelocityController.
`double`	`getActualVelocity()` G etsthe actual, current velocity.
`double`	`getVelocity()` G ets the current target velocity.
`boolean`	`isEnabled()` Is the VelocityController currently enabled1
`void`	`setAngleGains(double pAngle, double iAngle, double dAngle)` 2 asedon gyro angles TODO - javadoc
`void`	`setGains(double p, double i, double d)` ! econfigures the gains on the PIDController.
`void`	`setVelocity(double velocity)` 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)
```

InitialiWes a newVelocityController.

Parameters:

Parameters:

`p` - The proportional term for the PIDController.

`i` - The integral term for the PIDController.

`d` - The derivative term for the PIDController.

`encoder` - The encoder to use for feedback.

`output` - The PIDOutput to control. Usually some sort of motor.

Method Detail

getVelocity

```
public double getVelocity()
```

Gets the current target velocity.

Returns:

The current target velocity.

getActualVelocity

```
public double getActualVelocity()
```

Gets the actual, current velocity.

Returns:

The actual, current velocity.

setVelocity

```
public void setVelocity(double velocity)
```

Sets the target velocity.

Parameters:

`velocity` - The target velocity to set.

setGains

```
public void setGains(double p,
            double i,
            double d)
```

Reconfigures the gains on the PIDController.

Parameters:

`p` - The proportional term for the PIDController.

`i` - The integral term for the PIDController.

`d` - The derivative term for the PIDController.

setAngleGains

```
public void setAngleGains(double pAngle,
                 double iAngle,
                 double dAngle)
```

Based on gyro angles TODO - javadoc

Parameters:

`p` - The

`i` - The

`d` - The

enable

```
public void enable()
```

(nables theVelocityController.

disable

```
public void disable()
```

Disables the VelocityController.

isDnabled

```
public boolean isEnabled()
```

Is the VelocityController currently enabled1

Returns:

- hethepr 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 D esription
`Gyro360(AnalogChannel channel)` Initializes a new Gyro360 on the specified AnalogChannel.
`Gyro360(int port)` Initializes a new Gyro360 on the specified PW M port.
`Gyro360(int slot, int port)` Initializes a new Gyro360 on the specified PW M port on the spefied module port.

Method Summary

Methods

Modifier and Type	Method and D esription
`double`	`getAngle()` Gets the angle of the gyro, constrained to 360 degrees.
`double`	`pidGet()` 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.