PIDController Class Reference

Class that implements a PID Controller. More...

```
#include <PIDController.h>
```

Public Member Functions

	PIDController (float Kp, float Ki, float Kd) Constructor which creates and initializes a motor object. More...
void	**setKp** (float Kp) Updates the proportional gain. More...
void	**setKi** (float Ki) Updates the integral gain. More...
void	**setKd** (float Kd) Updates the derivative gain. More...
void	**setSetpoint** (int32_t setpoint) Updates the setpoint. More...
void	**setLoSat** (int8_t loSat) Updates the low saturation limit. More...
void	**setHiSat** (int8_t hiSat) Updates the high saturation limit. More...
void	**calc** (int32_t measurement) Update the previous error, the error, the integral, and the derivative. More...
float	**get** () Update the saturation status and return the PID output. More...
void	**reset** () Reset the calculated integral and derivative to zero. More...

Detailed Description

Class that implements a PID Controller.

This class allows the user to implement a PID Controller. It is given a setpoint, low saturation, high saturation, and Kp, Ki, and Kd gain values. In addition to providing PID output, the class accounts for integral windup by disabling the integral term if the output is determined as saturated. The class can also serve as a P-Only, PI, or PD controller by setting the appropriate gains to 0, although this does result in some unnecessary calculations.

Constructor & Destructor Documentation

◆ PIDController()

```
PIDController::PIDController ( float  Kp,
                               float  Ki,
                               float  Kd
                             )
```

Constructor which creates and initializes a motor object.

This constructor creates a PID controller object with a given proportional gain, integral gain, and derivative gain. If a P-Only, PI, or PD controller is desired, set the other gains to zero.

Parameters

Kp	The proportional gain for the controller.
Ki	The integral gain for the controller.
Kd	The derivative gain for the controller.

Member Function Documentation

◆ calc()

```
void PIDController::calc ( int32_t  measurement )
```

Update the previous error, the error, the integral, and the derivative.

This function updates the previous error, the error, the integral, and the derivative values. These values can be used to calculate the PID output and saturation status, but this calculation is left until the **get()** function is called to minimize the time spent in the interrupt service routine.

◆ get()

float PIDController::get ( )

Update the saturation status and return the PID output.

This function calculatest the PID output. It then compares the calculated output to the low saturation limit and the high saturation limit. If the PID output is outside of the saturation limits, the saturation status is set to true, else it is set to false. The PID output is then returned.

Returns

get The PID output as a float.

◆ reset()

void PIDController::reset ( )

Reset the calculated integral and derivative to zero.

This function resets the calculated integral and derivative to zero so that the PID controller can be re-entered without an affect from old data.

◆ setHiSat()

void PIDController::setHiSat (int8_t hiSat)

Updates the high saturation limit.

This function updates the high saturation limit to a new value during use.

Parameters

hiSat The new high saturation limit for the controller.

◆ setKd()

void PIDController::setKd (float Kd)

Updates the derivative gain.

This function updates the derivative gain to a new value during use.

Parameters

Kd The new derivative gain for the controller.

◆ setKi()

void PIDController::setKi (float Ki)

Updates the integral gain.

This function updates the integral gain to a new value during use.

Parameters

Ki The new integral gain for the controller.

◆ setKp()

void PIDController::setKp (float Kp)

Updates the proportional gain.

This function updates the proportional gain to a new value during use.

Parameters

Kp The new proportional gain for the controller.

◆ setLoSat()

void PIDController::setLoSat (int8_t loSat)

Updates the low saturation limit.

This function updates the low saturation limit to a new value during use.

Parameters

loSat	The new low saturation limit for the controller.

◆ setSetpoint()

void PIDController::setSetpoint (int32_t setpoint)

Updates the setpoint.

This function updates the setpoint to a new value during use.

Parameters

setpoint	The new proportional gain for the controller.

The documentation for this class was generated from the following files:

- **PIDController.h**
- **PIDController.cpp**