

# FRC 2018 Software Documentation

Team 5572: The ROSBOTS



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Namespace Index</b>   | <b>1</b> |
| 1.1      | Namespace List . . . . .                                       | 1        |
| <b>2</b> | <b>Class Index</b>   | <b>3</b> |
| 2.1      | Class List . . . . .   | 3        |
| <b>3</b> | <b>File Index</b>  | <b>5</b> |
| 3.1      | File List . . . . .  | 5        |
| <b>4</b> | <b>Namespace Documentation</b>                                 | <b>7</b> |
| 4.1      | drivetrain Namespace Reference . . . . .                       | 7        |
| 4.2      | math Namespace Reference . . . . .                             | 7        |
| 4.2.1    | Function Documentation . . . . .                               | 7        |
| 4.2.1.1  | smootherstep(double x) . . . . .                               | 7        |
| 4.2.1.2  | smoothstep(double x) . . . . .                                 | 8        |
| 4.2.1.3  | smoothstep_derivative(double x) . . . . .                      | 8        |
| 4.2.1.4  | smoothstep_derivative01(double x) . . . . .                    | 8        |
| 4.2.1.5  | wrapping_limit(double value, double min, double max) . . . . . | 8        |

|          |   |          |
|----------|---|----------|
| <b>5</b> | <b>Class Documentation</b>  | <b>9</b> |
| 5.1      | drivetrain::differential_curve Class Reference                          | 9        |
| 5.1.1    | Detailed Description  | 9        |
| 5.1.2    | Constructor & Destructor Documentation                                  | 9        |
| 5.1.2.1  | differential_curve(double x, double y, double w)                        | 9        |
| 5.1.2.2  | ~differential_curve()   | 9        |
| 5.1.3    | Member Function Documentation   | 10       |
| 5.1.3.1  | left()  | 10       |
| 5.1.3.2  | length()  | 10       |
| 5.1.3.3  | operator()()  | 10       |
| 5.2      | drivetrain::differential_drive Class Reference                          | 10       |
| 5.2.1    | Constructor & Destructor Documentation                                  | 10       |
| 5.2.1.1  | ~differential_drive()   | 10       |
| 5.2.2    | Member Function Documentation   | 11       |
| 5.2.2.1  | fromMotors(std::vector< unsigned > left, std::vector< unsigned > right) | 11       |
| 5.2.2.2  | fromMotors(std::vector< unsigned > left, std::vector< unsigned > right) | 11       |
| 5.2.2.3  | set(double, double)   | 11       |
| 5.3      | FRC5572Controller Class Reference                                       | 11       |
| 5.3.1    | Detailed Description  | 12       |
| 5.3.2    | Constructor & Destructor Documentation                                  | 12       |
| 5.3.2.1  | FRC5572Controller(int x)  | 12       |
| 5.3.2.2  | ~FRC5572Controller()  | 12       |
| 5.3.3    | Member Function Documentation   | 12       |
| 5.3.3.1  | A()   | 12       |
| 5.3.3.2  | B()   | 13       |
| 5.3.3.3  | back()  | 13       |
| 5.3.3.4  | L()   | 13       |
| 5.3.3.5  | LB()  | 13       |
| 5.3.3.6  | Lbutton()   | 13       |
| 5.3.3.7  | LT()  | 13       |
| 5.3.3.8  | POV()   | 13       |
| 5.3.3.9  | R()   | 13       |
| 5.3.3.10 | RB()  | 14       |
| 5.3.3.11 | Rbutton()   | 14       |
| 5.3.3.12 | RT()  | 14       |
| 5.3.3.13 | rumble(double, double)  | 14       |
| 5.3.3.14 | start()   | 14       |
| 5.3.3.15 | X()   | 14       |
| 5.3.3.16 | Y()   | 14       |

---

|          |  |           |
|----------|--|-----------|
| <b>6</b> | <b>File Documentation</b>                            | <b>15</b> |
| 6.1      | src/controller.h File Reference . . . . .            | 15        |
| 6.2      | src/drivetrain/drivetrain.h File Reference . . . . . | 15        |
| 6.3      | src/util/math.h File Reference . . . . .             | 16        |
| 6.3.1    | Macro Definition Documentation . . . . .             | 17        |
| 6.3.1.1  | PI . . . . .   | 17        |
|          | <b>Index</b>   | <b>19</b> |



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

|                            |       |                   |
|----------------------------|-------|-------------------|
| <a href="#">drivetrain</a> | ..... | <a href="#">7</a> |
| <a href="#">math</a>       | ..... | <a href="#">7</a> |





## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|  |    |
|--|----|
| <a href="#">drivetrain::differential_curve</a>                         |    |
| Instructions for following an arc given a differential drive . . . . . | 9  |
| <a href="#">drivetrain::differential_drive</a> . . . . .               | 10 |
| <a href="#">FRC5572Controller</a>                                      |    |
| Logitech Game Controller . . . . .                                     | 11 |



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

|  |    |
|--|----|
| src/ <a href="#">controller.h</a> . . . . .            | 15 |
| src/drivetrain/ <a href="#">drivetrain.h</a> . . . . . | 15 |
| src/util/ <a href="#">math.h</a> . . . . .             | 16 |



## Chapter 4

# Namespace Documentation

### 4.1 drivetrain Namespace Reference

#### Classes

- class [differential\\_curve](#)  
*Instructions for following an arc given a differential drive.*
- class [differential\\_drive](#)

### 4.2 math Namespace Reference

#### Functions

- double [wrapping\\_limit](#) (double value, double min, double max)  
*Enforces a wrapping limit on value.*
- double [smoothstep](#) (double x)  
*Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the derivative of the interpolation equals 0.*
- double [smoothstep\\_derivative](#) (double x)  
*Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0.*
- double [smoothstep\\_derivative01](#) (double x)  
*Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0, and the function reaches a maximum of 1, instead of 1.5 as in [smoothstep\\_derivative\(\)](#).*
- double [smootherstep](#) (double x)  
*Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the 1st and 2nd derivative of the interpolation equals 0.*

#### 4.2.1 Function Documentation

##### 4.2.1.1 `double math::smootherstep ( double x ) [inline]`

Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the 1st and 2nd derivative of the interpolation equals 0.

This is done using the function  $6x^5 - 15x^4 + 10x^3 | 0 \leq x \leq 1$ .

#### 4.2.1.2 `double math::smoothstep ( double x ) [inline]`

Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the derivative of the interpolation equals 0.

This is done using the function  $3x^2 - 2x^3 | 0 \leq x \leq 1$ .

#### 4.2.1.3 `double math::smoothstep_derivative ( double x ) [inline]`

Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0.

#### 4.2.1.4 `double math::smoothstep_derivative01 ( double x ) [inline]`

Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0, and the function reaches a maximum of 1, instead of 1.5 as in [smoothstep\\_derivative\(\)](#).

#### 4.2.1.5 `double math::wrapping_limit ( double value, double min, double max ) [inline]`

Enforces a wrapping limit on value.

Wrapping is a constraint in which a minimum is equal to a maximum, and values exceeding either limit "wraps" to the other extremum. An example is angles. The angles 0, and  $2\pi$  are equal, but if stored as a double, checking if they are equal will not produce the desired effect, so you may enforce a wrapping limit when checking for a value such as  $\pi$ .

##### Parameters

|              |                |
|--------------|----------------|
| <i>value</i> | value to limit |
| <i>min</i>   | minimum value  |
| <i>max</i>   | maximum value  |

## Chapter 5

# Class Documentation

### 5.1 drivetrain::differential\_curve Class Reference

Instructions for following an arc given a differential drive.

```
#include <drivetrain.h>
```

#### Public Member Functions

- [differential\\_curve](#) (double x, double y, double w)  
*Constructor with point and drivetrain width.*
- [~differential\\_curve](#) ()  
*Default destructor.*
- double & [operator\(\)](#) ()  
*Gets ratio of short edge to leading edge.*
- bool & [left](#) ()  
*Gets the side which is supposed to be the leading edge (true if left, false if right).*
- double & [length](#) ()  
*Gets overall length of leading edge.*

#### 5.1.1 Detailed Description

Instructions for following an arc given a differential drive.

#### 5.1.2 Constructor & Destructor Documentation

##### 5.1.2.1 drivetrain::differential\_curve::differential\_curve ( double x, double y, double w )

Constructor with point and drivetrain width.

##### 5.1.2.2 drivetrain::differential\_curve::~~differential\_curve ( ) [inline]

Default destructor.

### 5.1.3 Member Function Documentation

#### 5.1.3.1 `bool& drivetrain::differential_curve::left ( )` `[inline]`

Gets the side which is supposed to be the leading edge (true if left, false if right).

#### 5.1.3.2 `double& drivetrain::differential_curve::length ( )` `[inline]`

Gets overall length of leading edge.

#### 5.1.3.3 `double& drivetrain::differential_curve::operator() ( )` `[inline]`

Gets ratio of short edge to leading edge.

Always between 0 and 1, and doubles for position ratio and velocity ratio.

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

- [src/drivetrain/drivetrain.h](#)

## 5.2 `drivetrain::differential_drive` Class Reference

```
#include <drivetrain.h>
```

### Public Member Functions

- `~differential_drive ()`  
*Default destructor.*
- `void set (double, double)`  
*Set motor speeds.*
- `template<typename T >`  
`drivetrain::differential_drive fromMotors (std::vector< unsigned > left, std::vector< unsigned > right)`  
*creates differential drive given a motor-type and ids*

### Static Public Member Functions

- `template<typename T >`  
`static differential_drive fromMotors (std::vector< unsigned > left, std::vector< unsigned > right)`  
*creates differential drive given a motor-type and ids*

### 5.2.1 Constructor & Destructor Documentation

#### 5.2.1.1 `drivetrain::differential_drive::~~differential_drive ( )` `[inline]`

Default destructor.



## 5.2.2 Member Function Documentation

5.2.2.1 `template<typename T > static differential_drive drivetrain::differential_drive::fromMotors ( std::vector< unsigned > left, std::vector< unsigned > right ) [static]`

creates differential drive given a motor-type and ids

5.2.2.2 `template<typename T > drivetrain::differential_drive drivetrain::differential_drive::fromMotors ( std::vector< unsigned > left, std::vector< unsigned > right ) [inline]`

creates differential drive given a motor-type and ids

5.2.2.3 `void drivetrain::differential_drive::set ( double , double )`

Set motor speeds.

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

- [src/drivetrain/drivetrain.h](#)

## 5.3 FRC5572Controller Class Reference

Logitech Game Controller.

```
#include <controller.h>
```

### Public Member Functions

- [FRC5572Controller](#) (int x)  
*Constructor Sets up game controller and communication with the Driverstation.*
- [~FRC5572Controller](#) ()  
*Default destructor.*
- double [LT](#) ()  
*Returns the value of the left trigger The value returned will be between 0 and 1, with 0 being fully depressed and 1 being fully pressed.*
- bool [LB](#) ()  
*Returns the value of the left bumper The value returned will be true if the button is pressed, and false otherwise.*
- double [RT](#) ()  
*Returns the value of the right trigger The value returned will be between 0 and 1, with 0 being fully depressed and 1 being fully pressed.*
- bool [RB](#) ()  
*Returns the value of the right bumper The value returned will be true if the button is pressed, and false otherwise.*
- bool [X](#) ()  
*Returns the value of the blue X button The value returned will be true if the button is pressed, and false otherwise.*
- bool [Y](#) ()  
*Returns the value of the yellow Y button The value returned will be true if the button is pressed, and false otherwise.*
- bool [A](#) ()

- Returns the value of the green A button The value returned will be true if the button is pressed, and false otherwise.*

  - bool **B** ()

*Returns the value of the red B button The value returned will be true if the button is pressed, and false otherwise.*

  - std::pair< double, double > **L** ()

*Returns the values from the left joystick The value returned is a pair, with the first value being the x-coordinate of the joystick and the second value being the y-coordinate.*

  - std::pair< double, double > **R** ()

*Returns the values from the right joystick The value returned is a pair, with the first value being the x-coordinate of the joystick and the second value being the y-coordinate.*

  - int **POV** ()

*Returns the values from the D Pad.*

  - bool **start** ()

*Returns the value of the start button The value returned will be true if the button is pressed, and false otherwise.*

  - bool **back** ()

*Returns the value of the back button The value returned will be true if the button is pressed, and false otherwise.*

  - bool **Lbutton** ()

*Returns the value of the left button The value returned will be true if the button is pressed, and false otherwise.*

  - bool **Rbutton** ()

*Returns the value of the right button The value returned will be true if the button is pressed, and false otherwise.*

  - void **rumble** (double, double)

*Non-functional member function, which doesn't do anything.*

### 5.3.1 Detailed Description

Logitech Game Controller.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 FRC5572Controller::FRC5572Controller ( int x )

Constructor Sets up game controller and communication with the Driverstation.

##### Parameters

|   |   |
|---|---|
| x | The value associated with the controller. To determine this, open the drivestation and go to the controllers tab. Press any button on the controller. Whichever controller on the list highlights green is the controller, and the number next to it is the one you should insert here. |
|---|---|

#### 5.3.2.2 FRC5572Controller::~~FRC5572Controller ( )

Default destructor.

### 5.3.3 Member Function Documentation

#### 5.3.3.1 bool FRC5572Controller::A ( )

Returns the value of the green A button The value returned will be true if the button is pressed, and false otherwise.

#### 5.3.3.2 bool FRC5572Controller::B ( )

Returns the value of the red B button The value returned will be true if the button is pressed, and false otherwise.

#### 5.3.3.3 bool FRC5572Controller::back ( )

Returns the value of the back button The value returned will be true if the button is pressed, and false otherwise.

#### 5.3.3.4 std::pair<double, double> FRC5572Controller::L ( )

Returns the values from the left joystick The value returned is a pair, with the first value being the x-coordinate of the joystick and the second value being the y-coordinate.

The coordinates can be any value from -1 to 1, with 1 being fully up/right, -1 being fully down/left, and 0 being untouched.

#### 5.3.3.5 bool FRC5572Controller::LB ( )

Returns the value of the left bumper The value returned will be true if the button is pressed, and false otherwise.

#### 5.3.3.6 bool FRC5572Controller::Lbutton ( )

Returns the value of the left button The value returned will be true if the button is pressed, and false otherwise.

#### 5.3.3.7 double FRC5572Controller::LT ( )

Returns the value of the left trigger The value returned will be between 0 and 1, with 0 being fully depressed and 1 being fully pressed.

#### 5.3.3.8 int FRC5572Controller::POV ( )

Returns the values from the D Pad.

The value returned is an integer value which describes the location being pressed on the D Pad, with 0 being the upwards direction, and each other value may be taken as the degrees (from 0 to 360). If none of the buttons are being pressed, -1 is returned instead.

#### 5.3.3.9 std::pair<double, double> FRC5572Controller::R ( )

Returns the values from the right joystick The value returned is a pair, with the first value being the x-coordinate of the joystick and the second value being the y-coordinate.

The coordinates can be any value from -1 to 1, with 1 being fully up/right, -1 being fully down/left, and 0 being untouched.

**5.3.3.10 bool FRC5572Controller::RB ( )**

Returns the value of the right bumper The value returned will be true if the button is pressed, and false otherwise.

**5.3.3.11 bool FRC5572Controller::Rbutton ( )**

Returns the value of the right button The value returned will be true if the button is pressed, and false otherwise.

**5.3.3.12 double FRC5572Controller::RT ( )**

Returns the value of the right trigger The value returned will be between 0 and 1, with 0 being fully depressed and 1 being fully pressed.

**5.3.3.13 void FRC5572Controller::rumble ( double , double )**

Non-functional member function, which doesn't do anything.

It may do something in the future, with a different controller possibly.

**5.3.3.14 bool FRC5572Controller::start ( )**

Returns the value of the start button The value returned will be true if the button is pressed, and false otherwise.

**5.3.3.15 bool FRC5572Controller::X ( )**

Returns the value of the blue X button The value returned will be true if the button is pressed, and false otherwise.

**5.3.3.16 bool FRC5572Controller::Y ( )**

Returns the value of the yellow Y button The value returned will be true if the button is pressed, and false otherwise.

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

- [src/controller.h](#)

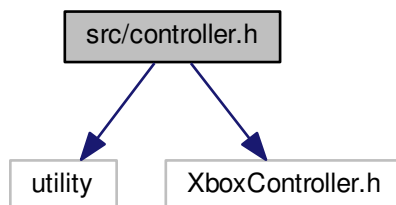
## Chapter 6

# File Documentation

### 6.1 src/controller.h File Reference

```
#include <utility>
#include <XboxController.h>
```

Include dependency graph for controller.h:



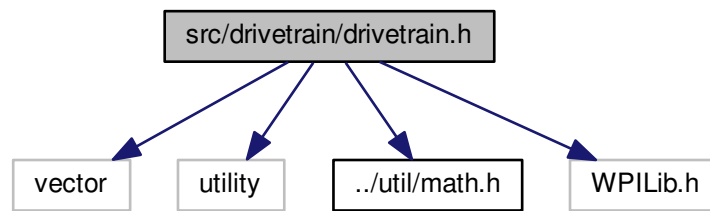
#### Classes

- class [FRC5572Controller](#)  
*Logitech Game Controller.*

### 6.2 src/drivetrain/drivetrain.h File Reference

```
#include <vector>
#include <utility>
#include "../util/math.h"
#include "WPILib.h"
```

Include dependency graph for drivetrain.h:



## Classes

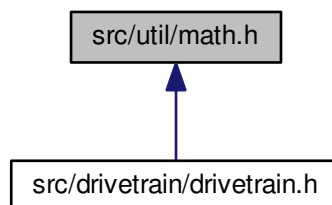
- class [drivetrain::differential\\_curve](#)  
*Instructions for following an arc given a differential drive.*
- class [drivetrain::differential\\_drive](#)

## Namespaces

- [drivetrain](#)

## 6.3 src/util/math.h File Reference

This graph shows which files directly or indirectly include this file:



## Namespaces

- [math](#)

## Macros

- `#define PI 3.141592654`

## Functions

- double `math::wrapping_limit` (double value, double min, double max)  
*Enforces a wrapping limit on value.*
- double `math::smoothstep` (double x)  
*Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the derivative of the interpolation equals 0.*
- double `math::smoothstep_derivative` (double x)  
*Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0.*
- double `math::smoothstep_derivative01` (double x)  
*Smooth interpolation function derivative Applies a smooth interpolation, such that at t=0 and t=1 the value equals 0, and the function reaches a maximum of 1, instead of 1.5 as in `smoothstep_derivative()`.*
- double `math::smootherstep` (double x)  
*Smooth interpolation function Applies a smooth interpolation, such that at t=0 and t=1 the 1st and 2nd derivative of the interpolation equals 0.*

### 6.3.1 Macro Definition Documentation

#### 6.3.1.1 `#define PI 3.141592654`





# Index

- ~FRC5572Controller
  - FRC5572Controller, [12](#)
- ~differential\_curve
  - drivetrain::differential\_curve, [9](#)
- ~differential\_drive
  - drivetrain::differential\_drive, [10](#)
- A
  - FRC5572Controller, [12](#)
- B
  - FRC5572Controller, [12](#)
- back
  - FRC5572Controller, [13](#)
- differential\_curve
  - drivetrain::differential\_curve, [9](#)
- drivetrain, [7](#)
- drivetrain::differential\_curve, [9](#)
  - ~differential\_curve, [9](#)
  - differential\_curve, [9](#)
  - left, [10](#)
  - length, [10](#)
  - operator(), [10](#)
- drivetrain::differential\_drive, [10](#)
  - ~differential\_drive, [10](#)
  - fromMotors, [11](#)
  - set, [11](#)
- FRC5572Controller, [11](#)
  - ~FRC5572Controller, [12](#)
  - A, [12](#)
  - B, [12](#)
  - back, [13](#)
  - FRC5572Controller, [12](#)
  - L, [13](#)
  - LB, [13](#)
  - Lbutton, [13](#)
  - LT, [13](#)
  - POV, [13](#)
  - R, [13](#)
  - RB, [13](#)
  - Rbutton, [14](#)
  - RT, [14](#)
  - rumble, [14](#)
  - start, [14](#)
  - X, [14](#)
  - Y, [14](#)
- fromMotors
  - drivetrain::differential\_drive, [11](#)

- L
  - FRC5572Controller, [13](#)
- LB
  - FRC5572Controller, [13](#)
- Lbutton
  - FRC5572Controller, [13](#)
- left
  - drivetrain::differential\_curve, [10](#)
- length
  - drivetrain::differential\_curve, [10](#)
- LT
  - FRC5572Controller, [13](#)
- math, [7](#)
  - smootherstep, [7](#)
  - smoothstep, [7](#)
  - smoothstep\_derivative, [8](#)
  - smoothstep\_derivative01, [8](#)
  - wrapping\_limit, [8](#)
- math.h
  - PI, [17](#)
- operator()
  - drivetrain::differential\_curve, [10](#)
- POV
  - FRC5572Controller, [13](#)
- PI
  - math.h, [17](#)
- R
  - FRC5572Controller, [13](#)
- RB
  - FRC5572Controller, [13](#)
- Rbutton
  - FRC5572Controller, [14](#)
- RT
  - FRC5572Controller, [14](#)
- rumble
  - FRC5572Controller, [14](#)
- set
  - drivetrain::differential\_drive, [11](#)
- smootherstep
  - math, [7](#)
- smoothstep
  - math, [7](#)
- smoothstep\_derivative
  - math, [8](#)
- smoothstep\_derivative01
  - math, [8](#)

src/controller.h, [15](#)  
src/drivetrain/drivetrain.h, [15](#)  
src/util/math.h, [16](#)  
start  
    FRC5572Controller, [14](#)  
  
wrapping\_limit  
    math, [8](#)  
  
X  
    FRC5572Controller, [14](#)  
  
Y  
    FRC5572Controller, [14](#)