Team Godspeed

Generated by Doxygen 1.8.20

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Namespace Documentation

3.1 godspeed Namespace Reference

The primary namespace for the project.

Namespaces

- Binder

    A namespace containing functions for binding value producing functions to value consuming functions.
- framework

    Contains core classes used as building blocks for the rest of the project.
- inputs

    Contains all classes for input devices.
- outputs

    Contains all classes for output devices.

3.1.1 Detailed Description

The primary namespace for the project.

3.2 godspeed::Binder Namespace Reference

A namespace containing functions for binding value producing functions to value consuming functions.

Functions

- void Init ()

    Runs the binder update function on it's own thread.
- int Bind (double(∗source)(void), void(∗sink)(double))

    Bind a source function to a sink function.
- int Bind (double(∗source)(void), double(∗pipe)(double), void(∗sink)(double))

    Bind a source function to a sink function through a pipe function.
- void DisableBinding (int id)

    Disable a binding using the bindings ID.
- bool IsDisabled (int id)

    Check if a binding is disabled using the bindings ID.
- void Update ()

    calls all bindings that are not disabled. This does NOT need to be called manually.

Variables

- std::list< BINDING_TUPLE > bindings = std::list<BINDING_TUPLE>()

  A list of tuples. Each tuple represents a binding.
- std::list< int > disabled = std::list<int>()

  A list of disabled binding IDs.

3.2.1 Detailed Description

A namespace containing functions for binding value producing functions to value consuming functions.

There are 3 types of functions considered here, source functions which take no values and return a double, pipe functions which take a double and return a double, and sink functions which take a double and return nothing. The Bind function allows you to bind the output of a source function to a sink function (optionally through a pipe function). Then, once the Binder is initialized, it will go through and repeatedly call the sink functions with the value produced by the source functions as the input.

3.3 godspeed::framework Namespace Reference

Contains core classes used as building blocks for the rest of the project.

3.3.1 Detailed Description

Contains core classes used as building blocks for the rest of the project.

3.4 godspeed::inputs Namespace Reference

Contains all classes for input devices.

Namespaces

- BallStorage

  A namespace with a function for accessing the current number of balls stored in the robot.
- RangeFinderSuite

  A namespace containing functions to access the distance values from all the rangefinders.
- RemoteController

  A namespace containing functions for accessing all remote controller inputs.

Classes

- class PathScript

  A class for creating scripted paths for the robot to take.

3.4.1 Detailed Description

Contains all classes for input devices.

3.5 godspeed::inputs::BallStorage Namespace Reference

A namespace with a function for accessing the current number of balls stored in the robot.

Functions

- void inc ()

  Increment the ball count. This does NOT need to be called manually.
- void dec ()

  Decrement the ball count. This does NOT need to be called manually.
- double BallCount ()

  Returns the current ball count.
- void Init ()

  Setup the ball storage counter to track balls.

Variables

- int BallCounter = 0

  A variable for tracking the ball count.

3.5.1 Detailed Description

A namespace with a function for accessing the current number of balls stored in the robot.

The Init() function must be called before the ball storage tracking can begin. That function attaches the inc and dec function as callbacks for when bumper A and bumper B are pressed (respectively).

3.6 godspeed::inputs::RangeFinderSuite Namespace Reference

A namespace containing functions to access the distance values from all the rangefinders.

Functions

- double RangeFinder ()

  returns the distance value of Range Finder 1

3.6.1 Detailed Description

A namespace containing functions to access the distance values from all the rangefinders.

3.7 godspeed::inputs::RemoteController Namespace Reference

A namespace containing functions for accessing all remote controller inputs.

Functions

- double **UpButton** ()
- double **DownButton** ()
- double **RightButton** ()
- double **LeftButton** ()
- double **XButton** ()
- double **YButton** ()
- double **AButton** ()
- double **BButton** ()
- double **RightTrigger** ()
- double **RightBumper** ()
- double **LeftTrigger** ()
- double **LeftBumper** ()
- double **LeftStickX** ()
- double **LeftStickY** ()
- double **RightStickX** ()
- double **RightStickY** ()

3.7.1 Detailed Description

A namespace containing functions for accessing all remote controller inputs.

3.8 godspeed::outputs Namespace Reference

Contains all classes for output devices.

Namespaces

- BallCollector

  A namespace containing functions for controlling the ball collector.
- BallScorer

  A namespace containing functions for controlling the ball scorer.
- OmniDrive3Wheel

  A namespace with functions for controlling the drive train.
- OutputUtilities

  A namespace for utility functions used by output classes.

3.8.1 Detailed Description

Contains all classes for output devices.

3.9 godspeed::outputs::BallCollector Namespace Reference

A namespace containing functions for controlling the ball collector.

Functions

- void TreadSpeed (double speed)

  Sets the speed of the two collector treads.

3.9.1 Detailed Description

A namespace containing functions for controlling the ball collector.

This namespace contains a single function that corresponds to both collector arm treads.

3.10 godspeed::outputs::BallScorer Namespace Reference

A namespace containing functions for controlling the ball scorer.

Functions

- void TreadSpeed (double speed)

  Sets the speed of the center tread.
- void ExpanderPosition (double angleDeg)

  Sets the angular position of the motor that extends the ball guide.

3.10.1 Detailed Description

A namespace containing functions for controlling the ball scorer.

This class contains two functions, one for the center tread and one for the ball guide expander.

3.11 godspeed::outputs::OmniDrive3Wheel Namespace Reference

A namespace with functions for controlling the drive train.

Functions

- void SetVelocity (double x, double y, double a)

  Set the X, Y, and angular velocities of the drivetrain.
- void XSpeed (double x)

  Sets the x-speed of the drivetrain.
- void YSpeed (double y)

  Sets the y-speed of the drivetrain.
- void AngleSpeed (double a)

  Sets the angular speed of the drivetrain.

Variables

- double XSpeedVar

  A variable to track the current desired x-speed.
- double YSpeedVar

  A variable to track the current desired y-speed.
- double AngleSpeedVar

  A variable to track the current desired angular speed.

3.11.1 Detailed Description

A namespace with functions for controlling the drive train.

3.12 godspeed::outputs::OutputUtilities Namespace Reference

A namespace for utility functions used by output classes.

Functions

- void setMotorSpeed (double motorSpeed, motor &m)

  Sets the speed of a motor.

3.12.1 Detailed Description

A namespace for utility functions used by output classes.

3.12.2 Function Documentation

3.12.2.1 setMotorSpeed()

```
void godspeed::outputs::OutputUtilities::setMotorSpeed (
            double motorSpeed,
            motor & m )
```

Sets the speed of a motor.

Takes a value -1 to 1 and sets the velocity appropriately. Has a "dead zone" of -0.1 to 0.1 which it treats as equal to zero (no movement)

Chapter 4

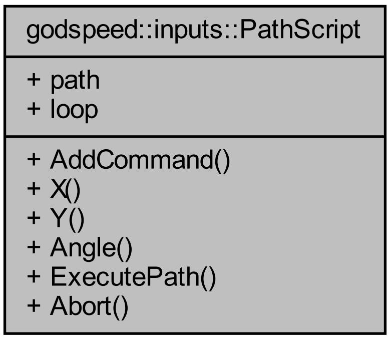
Class Documentation

4.1 godspeed::inputs::PathScript Class Reference

A class for creating scripted paths for the robot to take.

```
#include <path-script.h>
```

Collaboration diagram for godspeed::inputs::PathScript:



Public Member Functions

- void AddCommand (double x, double y, double a, double duration)

  Adds a command to the path script.

Static Public Member Functions

- static double **X** ()
- static double **Y** ()
- static double **Angle** ()
- static void ExecutePath (PathScript &path)

  Starts execution of the given path.
- static void Abort ()

  Flags the current path to stop execution once the current update cycle is finished.

Public Attributes

- std::list< COMMAND_TUPLE > path

  List of commands of form x-speed, y-speed, angle-speed, duration.
- bool loop

  Set this to true if you wish the path script to repeat after finishing.

4.1.1 Detailed Description

A class for creating scripted paths for the robot to take.

The paths are scripted using "commands" which are tuples consisting of an x-speed, a y-speed, an angular speed, and a duration. First create an instance of PathScript, then add all commands you want, then call PathScript::ExecutePath() passing in the PathScript you just created as the argument. This will immediately begin execution of the path. Path execution proceeds by recursive time-delayed updates which update variables, then whatever is bound the X(), Y(), and Angle() will have access to the updated variables. So the udpates are asyncronous from the actual movement changes and may not be deterministic.

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

- C:/Users/anzel/source/repos/team-godspeed/include/inputs/path-script.h

Index