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

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 godspeed Namespace Reference

The primary namespace for the project.

Namespaces

- 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.

4.1.1 Detailed Description

The primary namespace for the project.

4.2 godspeed::framework Namespace Reference

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

Classes

- class ActiveObject

  Abstract class for an object with it's own control thread.
- class DataSink

  An class representing a data sink of an output device or internal object.
- class DataSinkB

  An class representing a data sink of an output device or internal object.
- class DataSinkD

  An class representing a data sink of an output device or internal object.
- class DataSource

  An class representing a data source of an input device or internal object.
- class Event

  A class for implementing observer patterns.

Functions

- void ActiveObject_upd (void ∗args)

4.2.1 Detailed Description

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

4.2.2 Function Documentation

4.2.2.1 ActiveObject_upd()

```
void godspeed::framework::ActiveObject_upd (
             void * args )
```

The actual function handed to the thread to run

This function loops through the overrided update() method as long as isRunning() returns true. The parameter it takes is a reference to the ActiveObject. This is defined outside the class because you cannot run a class method on the thread object.

4.3 godspeed::inputs Namespace Reference

Contains all classes for input devices.

Classes

- class PathScript
- class RemoteController

  A class containing data source objects corresponding to controller inputs.

4.3.1 Detailed Description

Contains all classes for input devices.

4.4 godspeed::outputs Namespace Reference

Contains all classes for output devices.

Namespaces

- outputUtilities

  A namespace for utility functions used by output classes.

Classes

- class BallCollector

  A class containing data sink objects corresponding to the ball collector.
- class BallScorer

  A class containing data sink objects corresponding to the ball scorer.
- class OmniDrive3Wheel

  A class containing data sink objects corresponding to the drivetrain.

4.4.1 Detailed Description

Contains all classes for output devices.

4.5 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.

4.5.1 Detailed Description

A namespace for utility functions used by output classes.

4.5.2 Function Documentation

4.5.2.1 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 5

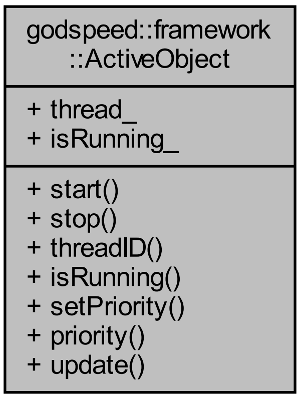
Class Documentation

5.1 godspeed::framework::ActiveObject Class Reference

Abstract class for an object with it's own control thread.

```
#include <active-object.h>
```

Collaboration diagram for godspeed::framework::ActiveObject:



Public Member Functions

- virtual void start ()

  Starts the active object.
- void stop ()

  Stops the active object.
- int32_t threadID ()

  Gets the ID of the current thread.

- bool isRunning ()

    Returns true if the object is currently running.
- void setPriority (int32_t priority)

    Sets the priority of the thread the object is running on.
- int32_t priority ()

    Gets the priority of the thread the object is running on.
- virtual void update ()

    The update function that is looped while the object is running.

Public Attributes

- vex::thread thread_

    The thread that the object runs on.
- bool isRunning_ = false

    The boolean used to control the thread.

5.1.1 Detailed Description

Abstract class for an object with it's own control thread.

Provides a framework for creating very simple active objects. Has a single thread that loops a function with no arguments or returns. This thread is created on running start and killed on running stop.

5.1.2 Member Function Documentation

5.1.2.1 setPriority()

```
void godspeed::framework::ActiveObject::setPriority (
            int32_t priority )
```

Sets the priority of the thread the object is running on.

Parameters

priority	The priority to set objects thread to

5.1.2.2 start()

```
void godspeed::framework::ActiveObject::start ( )  [virtual]
```

Starts the active object.

Creates a new thread that loops through the update function, and starts it if the object is not already running.

5.1.2.3 stop()

```
void godspeed::framework::ActiveObject::stop ( )
```

Stops the active object.

Kills the current thread the object is running on after it finishes the current loop through.

5.1.2.4 update()

```
void godspeed::framework::ActiveObject::update ( )  [virtual]
```

The update function that is looped while the object is running.

Must be overrided in a derivitive class. Note that to stop the thread the update function or an external object must call the stop() method. The update function will loop indefinitely if no stop is called.

5.1.3 Member Data Documentation

5.1.3.1 isRunning_

```
bool godspeed::framework::ActiveObject::isRunning_ = false
```

The boolean used to control the thread.

This value is set to true when a thread starts and is set to false by the stop() method. This signals the object to stop executing when it reaches the end of it's update function.

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

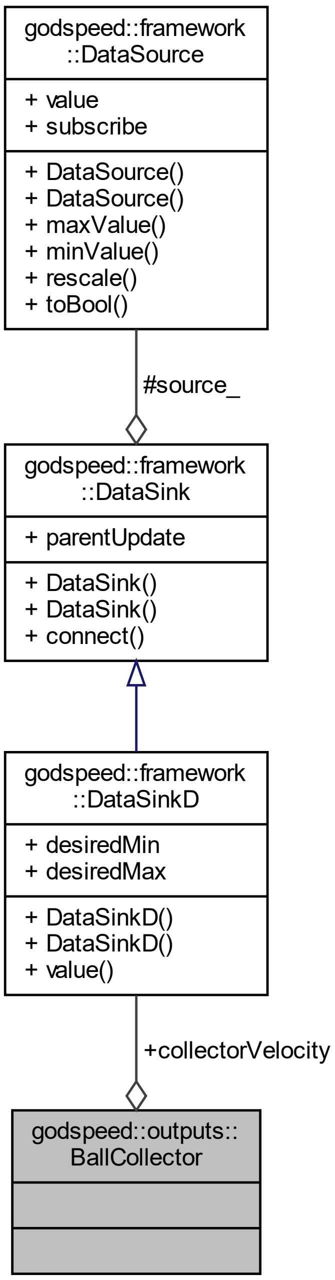
- C:/Users/anzel/source/repos/team-godspeed/include/framework/active-object.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/active-object.cpp

5.2 godspeed::outputs::BallCollector Class Reference

A class containing data sink objects corresponding to the ball collector.

```
#include <ball-collector.h>
```

Collaboration diagram for godspeed::outputs::BallCollector:



Static Public Attributes

- static DataSinkD collectorVelocity = DataSinkD(0, 1, update)

5.2.1 Detailed Description

A class containing data sink objects corresponding to the ball collector.

This class contains a single data sink object that corresponds to both collector arm treads. Currently does not allow for reverse direction.

5.2.2 Member Data Documentation

5.2.2.1 collectorVelocity

```
DataSinkD godspeed::outputs::BallCollector::collectorVelocity = DataSinkD(0, 1, update)  [static]
```

The data sink for the velocity of the collector arm treads

Does not allow for reverse direction

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

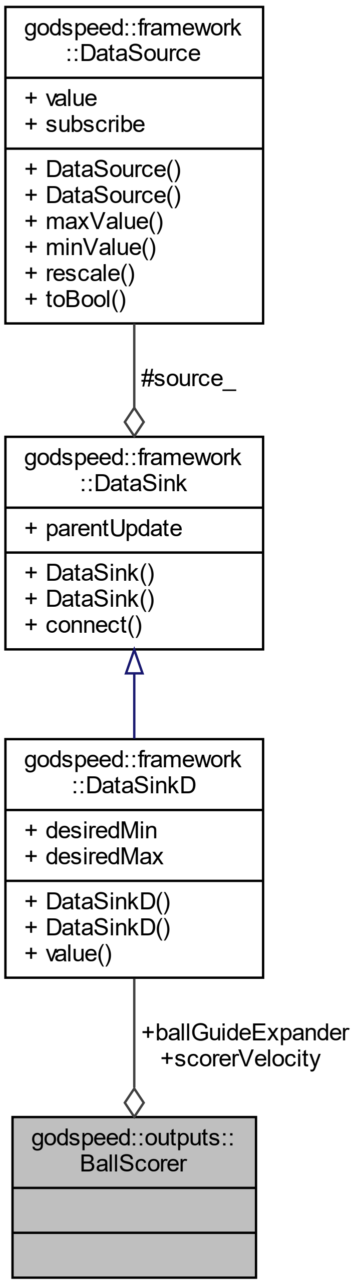
- C:/Users/anzel/source/repos/team-godspeed/include/outputs/ball-collector.h
- C:/Users/anzel/source/repos/team-godspeed/src/outputs/ball-collector.cpp

5.3 godspeed::outputs::BallScorer Class Reference

A class containing data sink objects corresponding to the ball scorer.

```
#include <ball-scorer.h>
```

Collaboration diagram for godspeed::outputs::BallScorer:



Static Public Attributes

- static DataSinkD scorerVelocity = DataSinkD(0, 1, update)

  The data sink for the velocity of the center (ball scorer) tread.
- static DataSinkD ballGuideExpander = DataSinkD(1, 1, update)

  The data sink for the ball guide expander.

5.3.1 Detailed Description

A class containing data sink objects corresponding to the ball scorer.

This class contains two data sink objects, one for the center tread and one for the ball guide expander. Currently does not allow reverse direction.

5.3.2 Member Data Documentation

5.3.2.1 ballGuideExpander

```
DataSinkD godspeed::outputs::BallScorer::ballGuideExpander = DataSinkD(1, 1, update)  [static]
```

The data sink for the ball guide expander.

The guide expander is a linear actuator that should initiate at the begining of a match. Currently the min and max of this object are both set to 1 as a temporary patch for not having support for setting a constant value data source.

5.3.2.2 scorerVelocity

```
DataSinkD godspeed::outputs::BallScorer::scorerVelocity = DataSinkD(0, 1, update)  [static]
```

The data sink for the velocity of the center (ball scorer) tread.

Does not allow for reverse direction

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

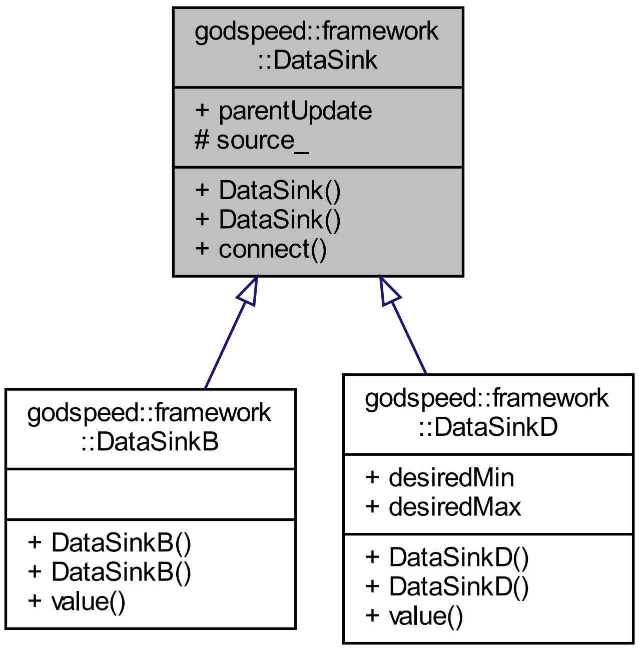
- C:/Users/anzel/source/repos/team-godspeed/include/outputs/ball-scorer.h
- C:/Users/anzel/source/repos/team-godspeed/src/outputs/ball-scorer.cpp

5.4 godspeed::framework::DataSink Class Reference

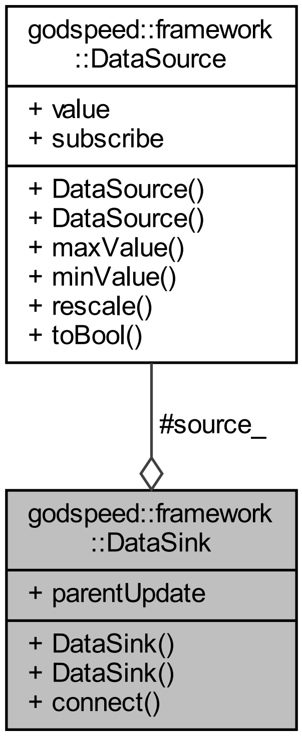
An class representing a data sink of an output device or internal object.

```
#include <data-sink.h>
```

Inheritance diagram for godspeed::framework::DataSink:



Collaboration diagram for godspeed::framework::DataSink:



Public Member Functions

- DataSink ()

    The default constructor.
- DataSink (void(∗parentUpd)(void))

    The primary constructor.
- void connect (DataSource &dat)

    Connects this data sink to a data source object.

Public Attributes

- void(∗ parentUpdate)(void)

    A reference to a function to call whenever the value of the connected data source changes.

Protected Attributes

- DataSource ∗ source_

    A reference to the connected data source object.

5.4.1 Detailed Description

An class representing a data sink of an output device or internal object.

A particular output device may require more than one input to control it, each of these inputs will be a data sink object. This is an incomplete class as it doesn't have a value getter. This class is extended by other data sinks with particular associated data types.

This class acts as an interface to objects that contain the DataSource object.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 DataSink()

```
godspeed::framework::DataSink::DataSink (
            void(*)(void) parentUpd )
```

The primary constructor.

Parameters

parentUpd	A function to call whenever the value of the connected data source changes

5.4.3 Member Function Documentation

5.4.3.1 connect()

```
void godspeed::framework::DataSink::connect (
            DataSource & dat )
```

Connects this data sink to a data source object.

Parameters

dat	the data source to connect to

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

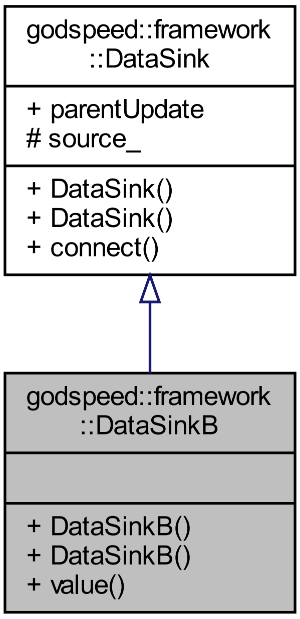
- C:/Users/anzel/source/repos/team-godspeed/include/framework/data-sink.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/data-sink.cpp

5.5 godspeed::framework::DataSinkB Class Reference

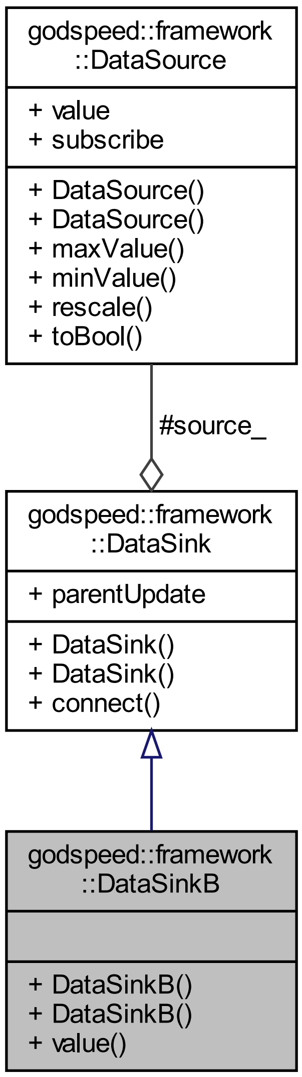
An class representing a data sink of an output device or internal object.

```
#include <data-sink-b.h>
```

Inheritance diagram for godspeed::framework::DataSinkB:



Collaboration diagram for godspeed::framework::DataSinkB:



Public Member Functions

- DataSinkB ()

  The default constructor.
- DataSinkB (void(∗parentUpd)(void))

  The primary constructor.
- bool value ()

  The value of the attached data source converted to a boolean.

Additional Inherited Members

5.5.1 Detailed Description

An class representing a data sink of an output device or internal object.

This is an implementation of a data source for the boolean data type.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DataSinkB()

```
godspeed::framework::DataSinkB::DataSinkB (
             void(*)(void) parentUpd )
```

The primary constructor.

Parameters

parentUpd	A function to call whenever the value of the connected data source changes

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

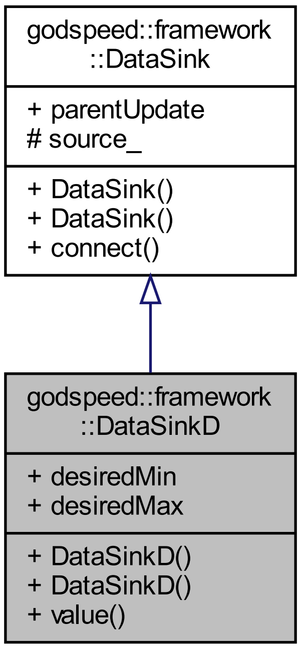
- C:/Users/anzel/source/repos/team-godspeed/include/framework/data-sink-b.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/data-sink-b.cpp

5.6 godspeed::framework::DataSinkD Class Reference

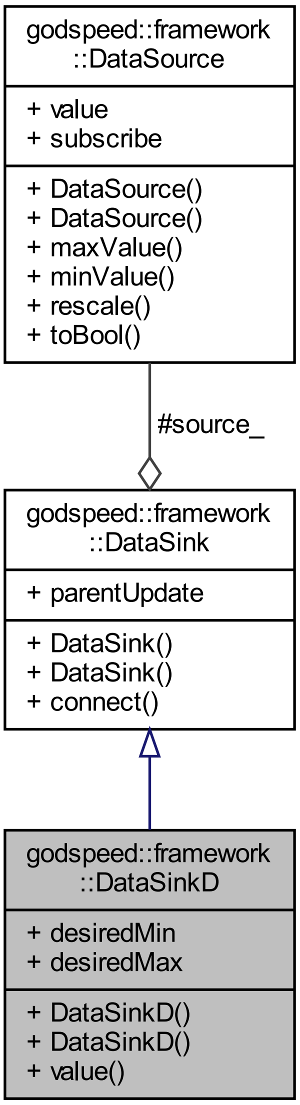
An class representing a data sink of an output device or internal object.

```
#include <data-sink-d.h>
```

Inheritance diagram for godspeed::framework::DataSinkD:



Collaboration diagram for godspeed::framework::DataSinkD:



Public Member Functions

- DataSinkD ()

  The default constructor.
- DataSinkD (double desMin, double desMax, void(∗parentUpd)(void))

  The primary constructor.
- double value ()

  The scaled value of the attached data source.

Public Attributes

- double desiredMin

  The desired minimum for the value() function to return.
- double desiredMax

  The desired maximum for the value() function to return.

Additional Inherited Members

5.6.1 Detailed Description

An class representing a data sink of an output device or internal object.

This is an implementation of a data source for the double data type.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 DataSinkD()

```
godspeed::framework::DataSinkD::DataSinkD (
          double desMin,
          double desMax,
          void(*)(void) parentUpd )
```

The primary constructor.

Parameters

desMin	The desired minimum for the value() function to return
desMax	The desired maximum for the value() function to return
parentUpd	A function to call whenever the value of the connected data source changes

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

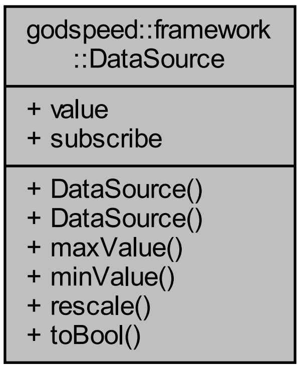
- C:/Users/anzel/source/repos/team-godspeed/include/framework/data-sink-d.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/data-sink-d.cpp

5.7 godspeed::framework::DataSource Class Reference

An class representing a data source of an input device or internal object.

```
#include <data-source.h>
```

Collaboration diagram for godspeed::framework::DataSource:



Public Member Functions

- DataSource ()

  The default constructor.
- DataSource (double min, double max, double(∗valFunc)(void), void(∗subscribeFunc)(void(∗callback)(void)))

  The primary constructor, initializes all necessary values.
- double maxValue ()

  The maximum value the data source will have.
- double minValue ()

  The minimum value the data source will have.

Static Public Member Functions

- static double rescale (double desiredMin, double desiredMax, DataSource &var)

  Rescales a data source value to be within a new min and max.
- static bool toBool (DataSource &var)

  Converts the data from a data source to a boolean.

Public Attributes

- double(∗ value)(void)

  The value getter for the data source.
- void(∗ subscribe)(void(∗callback)(void))

  Function to subscribe to value changes of the source.

5.7.1 Detailed Description

An class representing a data source of an input device or internal object.

A particular input device may produce more than one piece of data, each piece of data will be a data source. The data will need to be represented as a double, even if it is more accurately an integer or boolean.

This class acts as an interface to objects that contain the DataSink object.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 DataSource()

```
godspeed::framework::DataSource::DataSource (
            double min,
            double max,
            double(*)(void) valFunc,
            void(*)(void(*callback)(void)) subscribeFunc )
```

The primary constructor, initializes all necessary values.

Parameters

min	The minimum value the data source will ever output
max	The maximum value the data source will ever output
valFunc	Address of a function that will return the current value of the data source
subscribeFunc	Address of a function that will subscribe a given handler to value changes of the data source

5.7.3 Member Function Documentation

5.7.3.1 rescale()

```
double godspeed::framework::DataSource::rescale (
            double desiredMin,
            double desiredMax,
            DataSource & var )  [static]
```

Rescales a data source value to be within a new min and max.

This is used convert the data of a data source object into a form that a data sink can receive. Specifically this is used by the DataSinkD class.

Parameters

desiredMin	The desired minimum value for the data
desiredMax	The desired maximum value for the data
var	The data source object to rescale the value of

5.7.3.2 toBool()

```
bool godspeed::framework::DataSource::toBool (
            DataSource & var )  [static]
```

Converts the data from a data source to a boolean.

This is used convert the data of a data source object into a form that a data sink can receive. Specifically this is used by the DataSinkB class.

Returns true if the value of the data source is greater than half way between it's min and max. Otherwise returns false.

Parameters

var	The data source to convert the value of

5.7.4 Member Data Documentation

5.7.4.1 subscribe

```
void(* godspeed::framework::DataSource::subscribe) (void(*callback)(void))
```

Function to subscribe to value changes of the source.

The subscribe attribute should contain a reference to a function that can receive an event handler and subscribe said handlers to value changes of the data source. This means that it can be used with any event framework that allows for handlers with no arguments or returns.

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

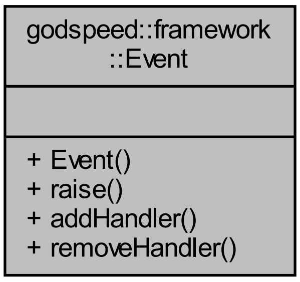
- C:/Users/anzel/source/repos/team-godspeed/include/framework/data-source.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/data-source.cpp

5.8 godspeed::framework::Event Class Reference

A class for implementing observer patterns.

```
#include <event.h>
```

Collaboration diagram for godspeed::framework::Event:



Public Member Functions

- Event ()

    The constructor for the event class.
- void raise ()

    Calling this method signifies that the event happened.
- Event addHandler (void(∗handler)(void))

    Add an event handler to the event.
- Event removeHandler (void(∗handler)(void))

    Remove a handler from the event.

5.8.1 Detailed Description

A class for implementing observer patterns.

The event class allows other objects to be notified when something happens. They do this by "subscribing" to the event by passing in a function that they wish to have called whenever that event happens. The event class will then go through it's list of functions and call each one of them whenever it is raised.

It is the job of the object creating the event to raise the event when appropriate. Other objects that wish to subscribe to the event must necessarily have access to the event object or to a public function that can receive an event handler function.

5.8.2 Member Function Documentation

5.8.2.1 addHandler()

```
Event godspeed::framework::Event::addHandler (
            void(*)(void) handler )
```

Add an event handler to the event.

Parameters

handler	A function with no arguments or returns that will be called when the event happens

5.8.2.2 raise()

```
void godspeed::framework::Event::raise ( )
```

Calling this method signifies that the event happened.

When called this method will call every event handler that has been added to the event.

5.8.2.3 removeHandler()

```
Event godspeed::framework::Event::removeHandler (
            void(*)(void) handler )
```

Remove a handler from the event.

To remove a handler you must have a reference to the original handler. Consequently this is usually only done by whatever object originally added the handler.

Parameters

handler	The handler function to remove

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

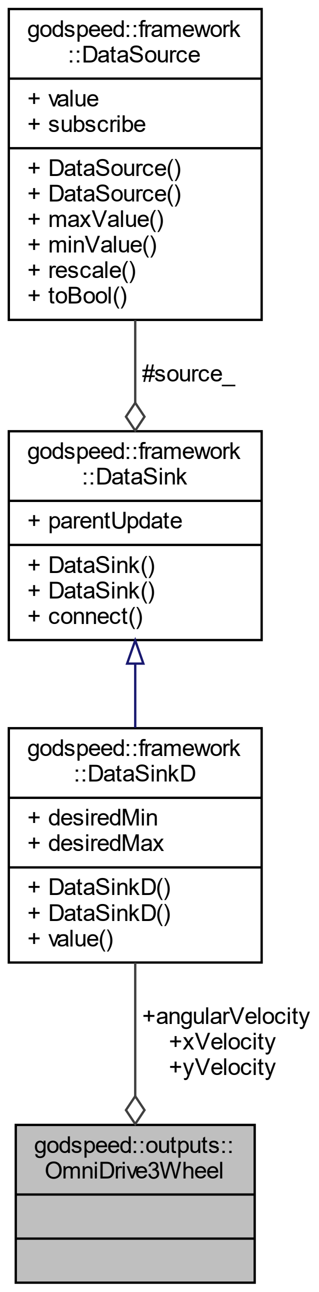
- C:/Users/anzel/source/repos/team-godspeed/include/framework/event.h
- C:/Users/anzel/source/repos/team-godspeed/src/framework/event.cpp

5.9 godspeed::outputs::OmniDrive3Wheel Class Reference

A class containing data sink objects corresponding to the drivetrain.

```
#include <omni-drive-3-wheel.h>
```

Collaboration diagram for godspeed::outputs::OmniDrive3Wheel:



Static Public Attributes

- static DataSinkD angularVelocity = DataSinkD(-1, 1, update)

  The data sink for the angular velocity of the drivetrain.
- static DataSinkD xVelocity = DataSinkD(-1, 1, update)

  The data sink for the x velocity of the drivetrain.
- static DataSinkD yVelocity = DataSinkD(-1, 1, update)

  The data sink for the y velocity of the drivetrain.

5.9.1 Detailed Description

A class containing data sink objects corresponding to the drivetrain.

This class contains data sink objects for the x velocity and y velocity of the drivetrain where positive y faces toward the front of the robot and positive x faces to the right.

5.9.2 Member Data Documentation

5.9.2.1 angularVelocity

```
DataSinkD godspeed::outputs::OmniDrive3Wheel::angularVelocity = DataSinkD(-1, 1, update)  [static]
```

The data sink for the angular velocity of the drivetrain.

Positive values indicate Clockwise rotation of the robot

5.9.2.2 xVelocity

```
DataSinkD godspeed::outputs::OmniDrive3Wheel::xVelocity = DataSinkD(-1, 1, update)  [static]
```

The data sink for the x velocity of the drivetrain.

Positive x faces to the right of the robot.

5.9.2.3 yVelocity

```
DataSinkD godspeed::outputs::OmniDrive3Wheel::yVelocity = DataSinkD(-1, 1, update)  [static]
```

The data sink for the y velocity of the drivetrain.

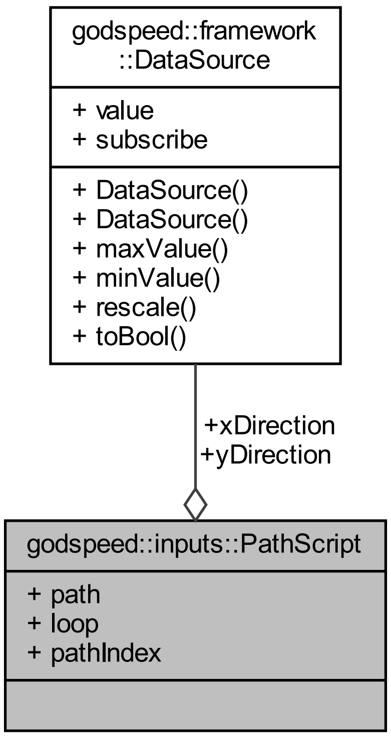
Positive y faces toward the front of the robot.

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

- C:/Users/anzel/source/repos/team-godspeed/include/outputs/omni-drive-3-wheel.h
- C:/Users/anzel/source/repos/team-godspeed/src/outputs/omni-drive-3-wheel.cpp

5.10 godspeed::inputs::PathScript Class Reference

Collaboration diagram for godspeed::inputs::PathScript:



Static Public Attributes

- static DataSource **xDirection**
- static DataSource **yDirection**
- static std::list< std::tuple< double, double, double > > **path**
- static bool **loop**
- static std::list< std::tuple< double, double, double > >::iterator **pathIndex** = path.begin()

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

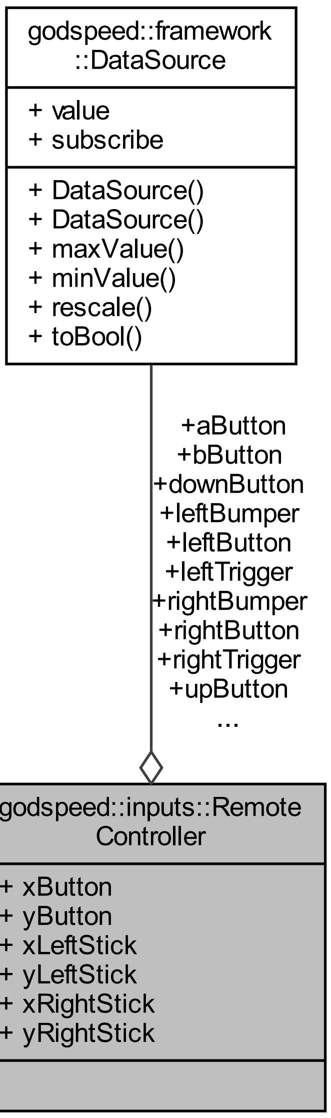
- C:/Users/anzel/source/repos/team-godspeed/include/inputs/path-script.h
- C:/Users/anzel/source/repos/team-godspeed/src/inputs/path-script.cpp

5.11 godspeed::inputs::RemoteController Class Reference

A class containing data source objects corresponding to controller inputs.

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

Collaboration diagram for godspeed::inputs::RemoteController:



Static Public Attributes

- static DataSource upButton = DataSource(0,1, upBtnVal, upBtnSubscribe)

  A data source corresponding to the state of the up button.

- static DataSource downButton = DataSource(0,1, downBtnVal, downBtnSubscribe)

  A data source corresponding to the state of the down button.
- static DataSource rightButton = DataSource(0,1, rightBtnVal, rightBtnSubscribe)

  A data source corresponding to the state of the right button.
- static DataSource leftButton = DataSource(0,1, leftBtnVal, leftBtnSubscribe)

  A data source corresponding to the state of the left button.
- static DataSource xButton = DataSource(0,1, xBtnVal, yBtnSubscribe)

  A data source corresponding to the state of the X button.
- static DataSource yButton = DataSource(0,1, yBtnVal, xBtnSubscribe)

  A data source corresponding to the state of the Y button.
- static DataSource aButton = DataSource(0,1, aBtnVal, aBtnSubscribe)

  A data source corresponding to the state of the A button.
- static DataSource bButton = DataSource(0,1, bBtnVal, bBtnSubscribe)

  A data source corresponding to the state of the B button.
- static DataSource rightTrigger = DataSource(0,1, rtVal, rtSubscribe)

  A data source corresponding to the state of the right trigger (R1)
- static DataSource rightBumper = DataSource(0,1, rbVal, rbSubscribe)

  A data source corresponding to the state of the right bumper (R2)
- static DataSource leftTrigger = DataSource(0,1, ltVal, ltSubscribe)

  A data source corresponding to the state of the left trigger (L1)
- static DataSource leftBumper = DataSource(0,1, lbVal, lbSubscribe)

  A data source corresponding to the state of the left bumper (L2)
- static DataSource xLeftStick = DataSource(-100,100, xlsVal, xlsSubscribe)

  A data source corresponding to the state of the left joy stick's X axis.
- static DataSource yLeftStick = DataSource(-100,100, ylsVal, ylsSubscribe)

  A data source corresponding to the state of the left joy stick's Y axis.
- static DataSource xRightStick = DataSource(-100,100, xrsVal, xrsSubscribe)

  A data source corresponding to the state of the right joy stick's X axis.
- static DataSource yRightStick = DataSource(-100,100, yrsVal, yrsSubscribe)

  A data source corresponding to the state of the right joy stick's Y axis.

5.11.1 Detailed Description

A class containing data source objects corresponding to controller inputs.

This class contains 16 data source objects. Each one corresponds to an individual input on the remote controller, such as the A-button or the the left stick X axis.

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

- C:/Users/anzel/source/repos/team-godspeed/include/inputs/remote-controller.h
- C:/Users/anzel/source/repos/team-godspeed/src/inputs/remote-controller.cpp

Index