Anti-Sway Capstone

1.0

Generated by Doxygen 1.11.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Angles
: A 2D Angle . . . . . ??

AntiSwayControlScheme
: Anti-Sway Mode Feedback Control Block . . . . . ??

Biquad
: Biquad . . . . . ??

DataFile_t
: Data File . . . . . ??

Differentiator
: Control Block: Differentiator . . . . . ??

Integrator
: Control Block: Integrator . . . . . ??

Positions
: A 2D Position . . . . . ??

ThreadResource
: Parameter for Threading Functions . . . . . ??

TrackingControlScheme
: Tracking Mode Feedback Control Block . . . . . ??

Velocities
: A 2D Velocity . . . . . ??

Chapter 2

File Index

2.1 File List

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

src/anti-sway.c
: Anti-Sway Control Law Implementation . . . . . ??

src/anti-sway.h
: Anti-Sway Control Law Header . . . . . ??

src/discrete-lib.c
: Discrete Control Law Implementation Library . . . . . ??

src/discrete-lib.h
: Discrete Control Law Implementation Library Header . . . . . ??

src/error.h
: Universal Error Library . . . . . ??

src/idle.c
: Idle Mode Implementation . . . . . ??

src/idle.h
: Idle Mode Header . . . . . ??

src/io.c
: Sensor/Actuator (Input/Output) Interfacing Library . . . . . ??

src/io.h
: Sensor/Actuator (Input/Output) Interfacing Library Header . . . . . ??

src/main.c
: Main File . . . . . ??

src/record.c
: Data Recording Interface . . . . . ??

src/record.h
: Data Recording Interface Header . . . . . ??

src/setup.c
: System Setup . . . . . ??

src/setup.h
: System Setup Header . . . . . ??

src/system.c
: System (Turing Machine) . . . . . ??

src/system.h
: System (Turing Machine) Header . . . . . ??

src/thread-lib.h
: Thread Library . . . . . ??

src/tracking.c
: Tracking Mode Control Law . . . . . ??

src/tracking.h
: Tracking Mode Control Law Header . . . . . ??

Chapter 3

Data Structure Documentation

3.1 Angles Struct Reference

A 2D Angle.

`#include <io.h>`

Data Fields

- Angle **x_angle**
- Angle **y_angle**

3.1.1 Detailed Description

A 2D Angle.

Defines the angle of the harness along both directions, in radians

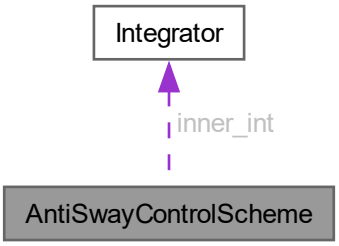
The documentation for this struct was generated from the following file:

- src/io.h

3.2 AntiSwayControlScheme Struct Reference

Anti-Sway Mode Feedback Control Block.

Collaboration diagram for AntiSwayControlScheme:



Data Fields

- Proportional **outer_feedback**
- Proportional **inner_prop**
- Integrator **inner_int**

3.2.1 Detailed Description

Anti-Sway Mode Feedback Control Block.

Represents the Inner and Outer Loop Elements

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

- src/anti-sway.c

3.3 Biquad Struct Reference

Biquad.

```
#include <discrete-lib.h>
```

Data Fields

- double **numerator** [3]
- double **denominator** [3]
- double **prev_input** [2]
- double **prev_output** [2]

3.3.1 Detailed Description

Biquad.

A struct representing a biquad

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

- src/discrete-lib.h

3.4 DataFile_t Struct Reference

Data File.

Data Fields

- MATFILE * **file**
- int **num_entries**
- char ** **entry_names**
- int **num_vals**
- int **vals_capacity**
- double ** **entry_values**

3.4.1 Detailed Description

Data File.

Internal Representation of a Data File

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

- src/record.c

3.5 Differentiator Struct Reference

Control Block: Differentiator.

```
#include <discrete-lib.h>
```

Data Fields

- Proportional **gain**
- double **prev_input**
- double **prev_output**

3.5.1 Detailed Description

Control Block: Differentiator.

A struct representing a derivative term

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

- src/discrete-lib.h

3.6 Integrator Struct Reference

Control Block: Integrator.

```
#include <discrete-lib.h>
```

Data Fields

- Proportional **gain**
- double **prev_input**
- double **prev_output**

3.6.1 Detailed Description

Control Block: Integrator.

A struct representing an integrator

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

- src/discrete-lib.h

3.7 Positions Struct Reference

A 2D Position.

`#include <io.h>`

Data Fields

- Position **x_pos**
- Position **y_pos**

3.7.1 Detailed Description

A 2D Position.

Defines the position of an object in 2D space, in meters

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

- src/io.h

3.8 ThreadResource Struct Reference

Parameter for Threading Functions.

`#include <thread-lib.h>`

Data Fields

- NiFpga_IrqContext **irq_context**
- NiFpga_Bool **irq_thread_rdy**

3.8.1 Detailed Description

Parameter for Threading Functions.

Represents a resource for a thread

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

- src/thread-lib.h

3.9 TrackingControlScheme Struct Reference

Tracking Mode Feedback Control Block.

Data Fields

- Proportional **combined_constants**
- Proportional **damping**

3.9.1 Detailed Description

Tracking Mode Feedback Control Block.

Represents the Inner and Outer Loop Elements

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

- src/tracking.c

3.10 Velocities Struct Reference

A 2D Velocity.

`#include <io.h>`

Data Fields

- Velocity **x_vel**
- Velocity **y_vel**

3.10.1 Detailed Description

A 2D Velocity.

Defines the velocity of an object in 2D space, in meters/second

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

- src/io.h

Chapter 4

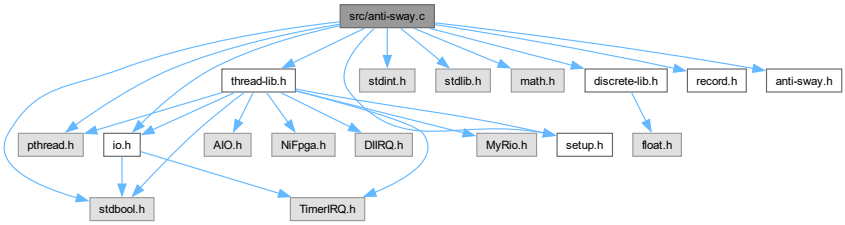
File Documentation

4.1 src/anti-sway.c File Reference

Anti-Sway Control Law Implementation.

```
#include <stdbool.h>
#include <pthread.h>
#include <stdint.h>
#include <stdlib.h>
#include <math.h>
#include "setup.h"
#include "io.h"
#include "thread-lib.h"
#include "discrete-lib.h"
#include "record.h"
#include "anti-sway.h"
```

Include dependency graph for anti-sway.c:



Data Structures

- struct AntiSwayControlScheme

  Anti-Sway Mode Feedback Control Block.

Macros

- #define **DATA_LEN** 20

Functions

- static void SetupScheme (AntiSwayControlScheme *scheme, Proportional K_p, Proportional K_↩ i, Proportional m)
- static void * AntiSwayModeThread (void *resource)
- static int AntiSwayControlLaw (Velocity vel_ref, Angle angle_input, Velocity vel_input, AntiSwayControlScheme *scheme, int(*SetVoltage)(Voltage voltage))
- int AntiSwayFork ()
- int AntiSwayJoin ()

Variables

- pthread_t **anti_sway_thread** = NULL
- ThreadResource **anti_sway_resource**
- static double **K_ptx** = 51.55550206284189
- static double **K_itx** = 33.28586146285062
- static double **K_pty** = 0.8*55.65965893434064
- static double **K_ity** = 0.8*31.59324977878787
- static AntiSwayControlScheme **x_control**
- static AntiSwayControlScheme **y_control**
- static int **error**
- static FileID_t **file** = -1
- static char * **data_file_name** = "anti-sway.mat"
- static char * data_names [DATA_LEN]
- static double **data** [DATA_LEN]
- static double * **data_buff** = data
- static int **id** = 1
- static double **t** = 0.0

4.1.1 Detailed Description

Anti-Sway Control Law Implementation.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.1.2 Function Documentation

4.1.2.1 AntiSwayControlLaw()

```
static int AntiSwayControlLaw (
            Velocity vel_ref,
            Angle angle_input,
            Velocity vel_input,
            AntiSwayControlScheme * scheme,
            int(* SetVoltage )(Voltage voltage))  [inline], [static]
```

Executes 1 timestep for the Anti-Sway Mode Control Law for its input to the plant

Parameters

vel_ref	The reference velocity for Anti-Sway Mode
angle_input	The measured rope angle for Anti-Sway Mode
vel_input	The measured velocity of the motor
scheme	A pointer to the AntiSwayControlScheme structure used to execute the control law
SetVoltage	The function that sets the voltage of the appropriate motor

Returns

0 upon success, negative otherwise

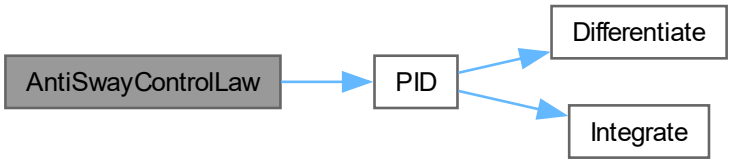
Precondition

scheme was not modified before use of this function

Postcondition

scheme is now updated with the input and outputs for the respective control scheme

Here is the call graph for this function:



4.1.2.2 AntiSwayFork()

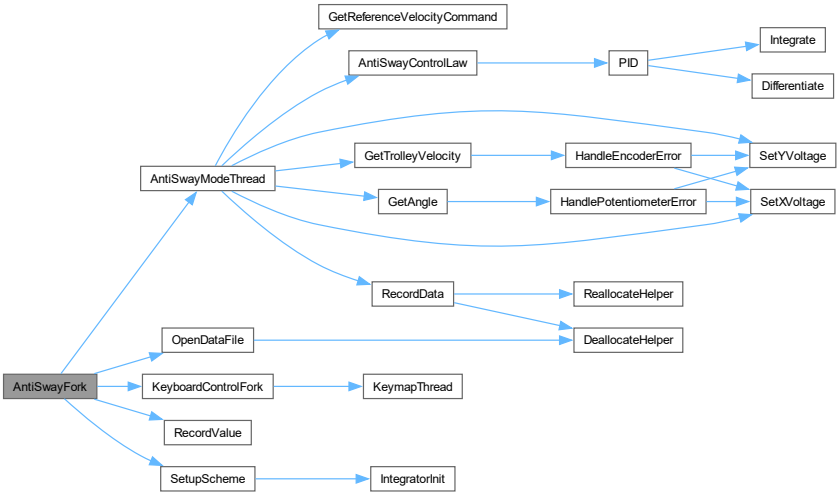
```
int AntiSwayFork ()
```

Executes Anti-Sway Mode (concurrently)

Postcondition

If its already running, does nothing

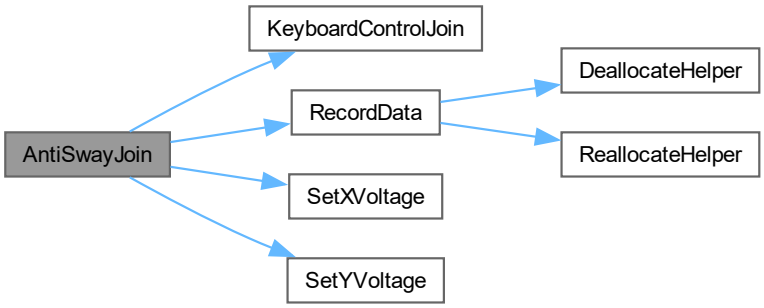
Here is the call graph for this function:



4.1.2.3 AntiSwayJoin()

```
int AntiSwayJoin ()
```

Stops Anti-Sway Mode (concurrent process) Here is the call graph for this function:



4.1.2.4 AntiSwayModeThread()

```
static void * AntiSwayModeThread (
            void * resource)  [static]
```

The Thread Function for Anti-Sway Mode

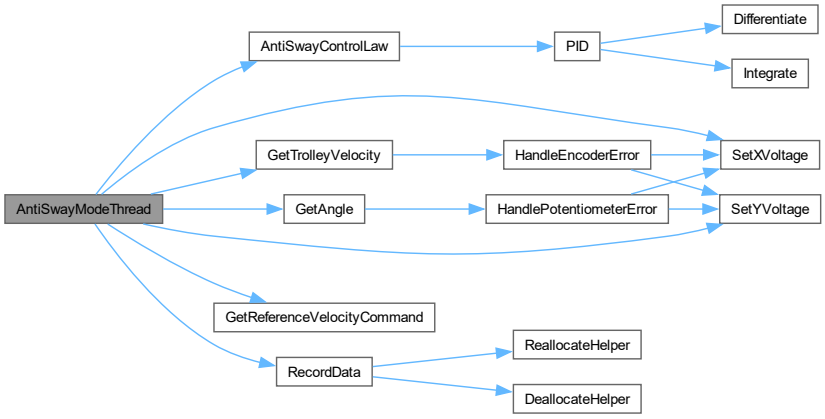
Parameters

resource	A pointer to a Resource sturcture for Tracking Mode

Returns

NULL

Here is the call graph for this function:



4.1.2.5 SetupScheme()

```
static void SetupScheme (
            AntiSwayControlScheme * scheme,
            Proportional K_p,
            Proportional K_i,
            Proportional m)  [inline], [static]
```

Sets up an AntiSwayControlScheme

Parameters

scheme	The scheme to setup
K_p	The proportional gain
K_i	The integral gain
m	The combined masses

Postcondition

scheme is now setup with zero initial conditions and proper constants

Here is the call graph for this function:

4.1.3 Variable Documentation

4.1.3.1 data_names

```
char* data_names[DATA_LEN]  [static]
```

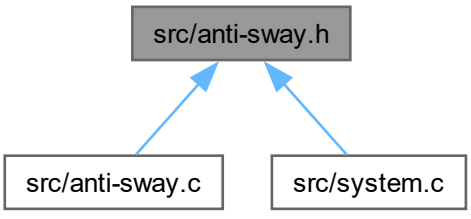
Initial value:

```
= {"id", "t",
                                        "vel_ref_x", "vel_ref_y",
                                        "angle_x", "angle_y",
                                        "trolley_vel_x", "trolley_vel_y",
                                        "vel_err_x", "voltage_x", "int_out_x", "Kp_x'", "Ki_x'", "loss_x",
                                        "vel_err_y", "voltage_y", "int_out_y", "Kp_y'", "Ki_y'", "loss_y"}
```

4.2 src/anti-sway.h File Reference

Anti-Sway Control Law Header.

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



Functions

- int AntiSwayFork ()
- int AntiSwayJoin ()

4.2.1 Detailed Description

Anti-Sway Control Law Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.2.2 Function Documentation

4.2.2.1 AntiSwayFork()

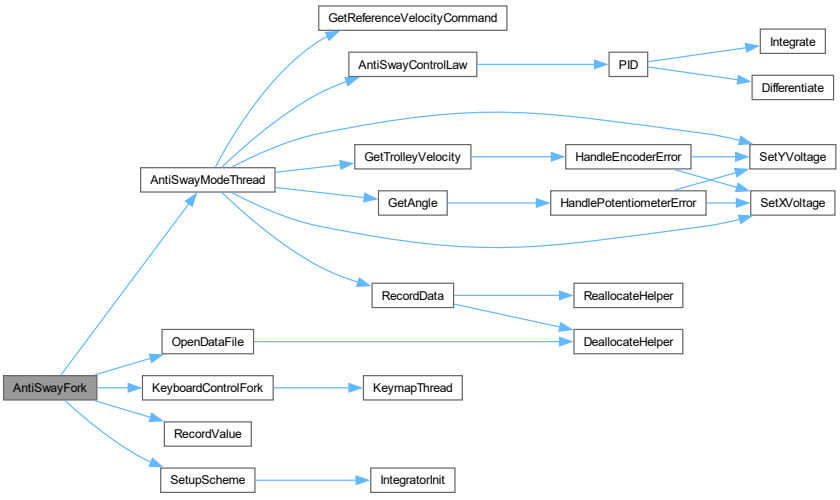
```
int AntiSwayFork ()
```

Executes Anti-Sway Mode (concurrently)

Postcondition

If its already running, does nothing

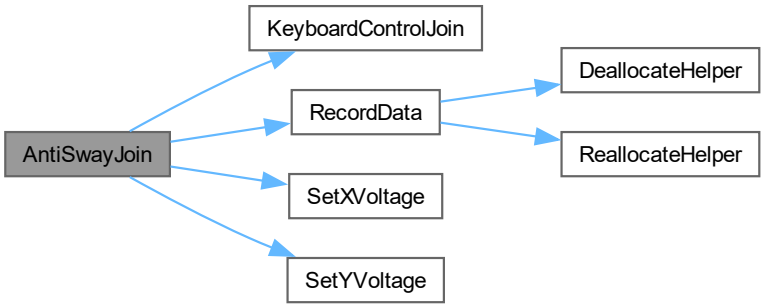
Here is the call graph for this function:



4.2.2.2 AntiSwayJoin()

```
int AntiSwayJoin ()
```

Stops Anti-Sway Mode (concurrent process) Here is the call graph for this function:



4.3 anti-sway.h

Go to the documentation of this file.

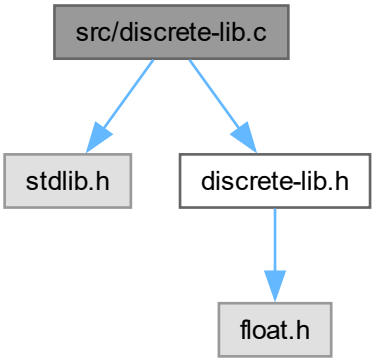
```
00001
00013 #ifndef ANTI_SWAY_H_
00014 #define ANTI_SWAY_H_
00015
00016 /* Execution-Dispatch Function */
00017
00018
00024 int AntiSwayFork();
00025
00029 int AntiSwayJoin();
00030
00031 #endif  // ANTI_SWAY_H_
```

4.4 src/discrete-lib.c File Reference

Discrete Control Law Implementation Library.

```
#include <stdlib.h>
#include "discrete-lib.h"
```

Include dependency graph for discrete-lib.c:



Macros

- #define **SATURATE**(val, lo, hi) val < lo ? lo : (val > hi ? hi : val)

Functions

- static double EvaluateBiquad (Biquad *sys, double input)
- void IntegratorInit (Proportional gain, double timestep, Integrator *result)
- void DifferentiatorInit (Proportional gain, double timestep, Differentiator *result)
- double Cascade (double input, Biquad sys[ ], int size, double lower_lim, double upper_lim)
- double Integrate (double input, Integrator *term, double lower_lim, double upper_lim)
- double Differentiate (double input, Differentiator *term, double lower_lim, double upper_lim)
- double PID (double input, Proportional *p, Integrator *i, Differentiator *d, double lower_lim, double upper_lim)

4.4.1 Detailed Description

Discrete Control Law Implementation Library.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.4.2 Function Documentation

4.4.2.1 Cascade()

```
double Cascade (
            double input,
            Biquad sys[],
            int size,
            double lower_lim,
            double upper_lim)  [inline]
```

Executes a dynamic, discrete time system by using its biquad decomposition.

Parameters

input	The input to the system
sys	The system, as an array of biquads
size	The size of sys
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the system given the input

Precondition

The input is the next sampled value of the input to the system

Postcondition

The system is updated with current/past calculated values

Here is the call graph for this function:

4.4.2.2 Differentiate()

```
double Differentiate (
            double input,
            Differentiator * term,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps a Differentiation

Parameters

input	The input to the differentiator
term	A pointer to an differentiator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the differentiator given the input

Precondition

The input is the next sampled value of the input to the system

Postcondition

term is updated with current/past calculated values

4.4.2.3 DifferentiatorInit()

```
void DifferentiatorInit (
            Proportional gain,
            double timestep,
            Differentiator * result)
```

Initializes a Differentiator

Parameters

gain	The gain to assign the differentiator
timestep	The timestep to approximate the differentiator
result	A return parameter, which becomes the differentiator with the gain and timestep

Returns

result, which will be an differentiator with a gain gain, and the timestep

4.4.2.4 EvaluateBiquad()

```
static double EvaluateBiquad (
            Biquad * sys,
            double input)  [inline], [static]
```

Evaluates a singular dynamic distrete time biquad within a system, which itself is a system.

Parameters

sys	The system
input	The system's input

Returns

The output of the system

Precondition

The input is the next sampled value of the input to the system

Postcondition

The system is updated with current/past calculated values

4.4.2.5 Integrate()

```
double Integrate (
            double input,
            Integrator * term,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps an Integration

Parameters

input	The input to the integrator
term	A pointer to an integrator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the integrator given the input

Precondition

The input is the next sampled value of the input to the system

Postcondition

term is updated with current/past calculated values

4.4.2.6 IntegratorInit()

```
void IntegratorInit (
            Proportional gain,
            double timestep,
            Integrator * result)
```

Initializes an Integrator

Parameters

gain	The gain to assign the integrator
timestep	The timestep to approximate the integrator
result	A return parameter, which becomes the integrator with the gain and timestep

Returns

result, which will be an integrator with a gain gain, and the timestep

4.4.2.7 PID()

```
double PID (
            double input,
            Proportional * p,
            Integrator * i,
            Differentiator * d,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps a PID Controller

Parameters

input	The input to the PID Controller
p	A pointer to the proportional term
i	A pointer to the integrator term
d	A pointer to the differentiator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the PID Controller given the input

Precondition

The input is the next sampled value of the input to the system

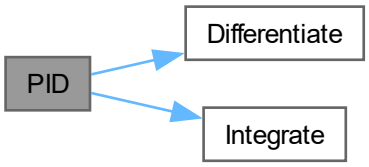
If p, i or d is NULL, then those NULL terms don't contribute

if p, i and d are all NULL, then the output is 0.0

Postcondition

i and d are updated with current/past calculated values

Here is the call graph for this function:

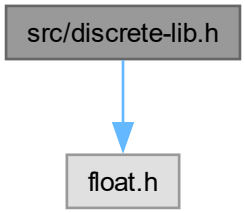


4.5 src/discrete-lib.h File Reference

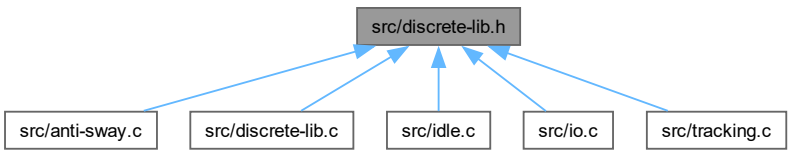
Discrete Control Law Implementation Library Header.

```
#include <float.h>
```

Include dependency graph for discrete-lib.h:



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



Data Structures

- struct Biquad

  Biquad.
- struct Integrator

  Control Block: Integrator.
- struct Differentiator

  Control Block: Differentiator.

Macros

- #define **POS_INF** DBL_MAX
- #define **NEG_INF** (-DBL_MAX)

Typedefs

- typedef float Proportional

  Control Block: Proportion.

Functions

- void IntegratorInit (Proportional gain, double timestep, Integrator ∗result)
- void DifferentiatorInit (Proportional gain, double timestep, Differentiator ∗result)
- double Cascade (double input, Biquad sys[ ], int size, double lower_lim, double upper_lim)
- double Integrate (double input, Integrator ∗term, double lower_lim, double upper_lim)
- double Differentiate (double input, Differentiator ∗term, double lower_lim, double upper_lim)
- double PID (double input, Proportional ∗p, Integrator ∗i, Differentiator ∗d, double lower_lim, double upper_lim)

4.5.1 Detailed Description

Discrete Control Law Implementation Library Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.5.2 Typedef Documentation

4.5.2.1 Proportional

```
typedef float Proportional
```

Control Block: Proportion.

A proportional constant

4.5.3 Function Documentation

4.5.3.1 Cascade()

```
double Cascade (
            double input,
            Biquad sys[],
            int size,
            double lower_lim,
            double upper_lim)  [inline]
```

Executes a dynamic, discrete time system by using its biquad decomposition.

Parameters

input	The input to the system
sys	The system, as an array of biquads
size	The size of sys
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the system given the input

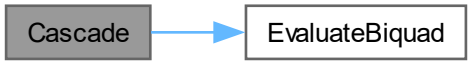
Precondition

The input is the next sampled value of the input to the system

Postcondition

The system is updated with current/past calculated values

Here is the call graph for this function:



4.5.3.2 Differentiate()

```
double Differentiate (
            double input,
            Differentiator * term,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps a Differentiation

Parameters

input	The input to the differentiator
term	A pointer to an differentiator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the differentiator given the input

Precondition

The input is the next sampled value of the input to the system

Postcondition

term is updated with current/past calculated values

4.5.3.3 DifferentiatorInit()

```
void DifferentiatorInit (
            Proportional gain,
            double timestep,
            Differentiator * result)
```

Initializes a Differentiator

Parameters

gain	The gain to assign the differentiator
timestep	The timestep to approximate the differentiator
result	A return parameter, which becomes the differentiator with the gain and timestep

Returns

result, which will be an differentiator with a gain gain, and the timestep

4.5.3.4 Integrate()

```
double Integrate (
            double input,
            Integrator * term,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps an Integration

Parameters

input	The input to the integrator
term	A pointer to an integrator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the integrator given the input

Precondition

The input is the next sampled value of the input to the system

Postcondition

term is updated with current/past calculated values

4.5.3.5 IntegratorInit()

```
void IntegratorInit (
            Proportional gain,
            double timestep,
            Integrator * result)
```

Initializes an Integrator

Parameters

gain	The gain to assign the integrator
timestep	The timestep to approximate the integrator
result	A return parameter, which becomes the integrator with the gain and timestep

Returns

result, which will be an integrator with a gain gain, and the timestep

4.5.3.6 PID()

```
double PID (
            double input,
            Proportional * p,
            Integrator * i,
            Differentiator * d,
            double lower_lim,
            double upper_lim)  [inline]
```

Timesteps a PID Controller

Parameters

input	The input to the PID Controller
p	A pointer to the proportional term
i	A pointer to the integrator term
d	A pointer to the differentiator term
lower_lim	The lower saturation limit of the system
upper_lim	The upper saturation limit of the system

Returns

The output of the PID Controller given the input

Precondition

The input is the next sampled value of the input to the system

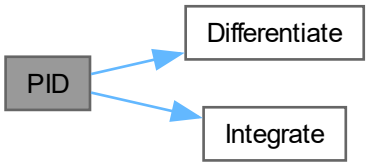
If p, i or d is NULL, then those NULL terms don't contribute

if p, i and d are all NULL, then the output is 0.0

Postcondition

i and d are updated with current/past calculated values

Here is the call graph for this function:



4.6 discrete-lib.h

Go to the documentation of this file.

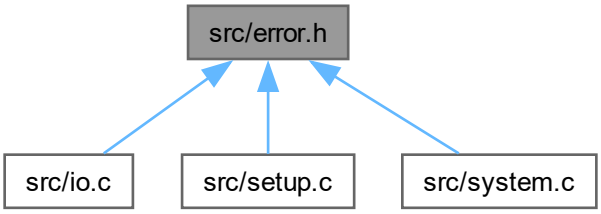
```
00001
00013 #ifndef DISCRETE_LIB_H_
00014 #define DISCRETE_LIB_H_
00015
00016 #include <float.h>
00017
00018 /* Non-saturation constants */
00019 #define POS_INF DBL_MAX
00020 #define NEG_INF (-DBL_MAX)
00021
00022
00023 /* Discrete-Time Data Structures */
00024
00025
```

```
00031 typedef struct {
00032     // The numerator coefficients, in decreasing order of time delays
00033     // (z^0, z^-1, z^-2)
00034     double numerator[3];
00035     // The denominator coefficients, in decreasing order of time delays
00036     // (z^0, z^-1, z^-2)
00037     double denominator[3];
00038     // The previous inputs, in increasing time delays
00039     // (z^-1, z^-2)
00040     double prev_input[2];
00041     // The previous outputs, in increasing time delays
00042     // (z^-1, z^-2)
00043     double prev_output[2];
00044 } Biquad;
00045
00051 typedef float Proportional;
00052
00058 typedef struct {
00059     Proportional gain;  // Accounts for timestep
00060     // Biquad integral =
00061     // {{1.0, 1.0, 0.0}, {2.0, -2.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}};
00062     double prev_input;
00063     double prev_output;
00064 } Integrator;
00065
00071 typedef struct {
00072     Proportional gain;  // Accounts for timestep
00073     // Biquad derivative =
00074     // {{2.0, -2.0, 0.0}, {1.0, 1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}};
00075     double prev_input;
00076     double prev_output;
00077 } Differentiator;
00078
00079
00080 /* Initialization Functions */
00081
00082
00094 void IntegratorInit(Proportional gain, double timestep, Integrator *result);
00095
00107 void DifferentiatorInit(Proportional gain,
00108                         double timestep,
00109                         Differentiator *result);
00110
00111
00112 /* Time-Stepping Functions */
00113
00114
00131 inline double Cascade(double input,
00132                       Biquad sys[],
00133                       int size,
00134                       double lower_lim,
00135                       double upper_lim);
00136
00151 inline double Integrate(double input,
00152                         Integrator *term,
00153                         double lower_lim,
00154                         double upper_lim);
00155
00170 inline double Differentiate(double input,
00171                             Differentiator *term,
00172                             double lower_lim,
00173                             double upper_lim);
00174
00193 inline double PID(double input,
00194                   Proportional *p,
00195                   Integrator *i,
00196                   Differentiator *d,
00197                   double lower_lim,
00198                   double upper_lim);
00199
00200 #endif  // DISCRETE_LIB_H_
```

4.7 src/error.h File Reference

Universal Error Library.

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



Macros

- #define **ENKWN** -1
- #define **EOTBD** -2
- #define **EVTYE** -3
- #define **ESTRN** -4
- #define **EENCR** -5

Variables

- int **u_error**

4.7.1 Detailed Description

Universal Error Library.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.8 error.h

Go to the documentation of this file.

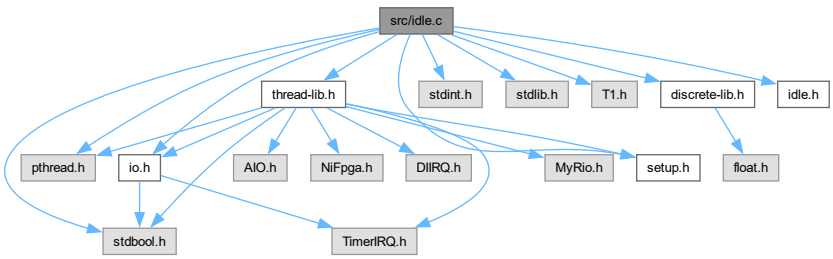
```
00001
00013 #ifndef ERROR_H_
00014 #define ERROR_H_
00015
00016 /* Universal Error Codes */
00017
00018
00019 /* Error Macro */
00020 // Ther universal error code
00021 extern int u_error;
00022
00023
00024 /* I/O Error Codes */
00025
00026 // Unknown Exception
00027 #define ENKWN -1
00028 // Out of Bounds Error
00029 #define EOTBD -2
00030 // Velocity Exceeded Error
00031 #define EVTYE -3
00032 // Unexpected Saturation Error
00033 #define ESTRN -4
00034 // Encoder Error
00035 #define EENCR -5
00036
00037
00038 #endif  // ERROR_H_
```

4.9 src/idle.c File Reference

Idle Mode Implementation.

```
#include <stdbool.h>
#include <pthread.h>
#include <stdint.h>
#include <stdlib.h>
#include "T1.h"
#include "setup.h"
#include "io.h"
#include "thread-lib.h"
#include "discrete-lib.h"
#include "idle.h"
```

Include dependency graph for idle.c:



Macros

- #define **DECIMAL_PRECISION** "3"
- #define **RAD_2_DEG**(value) value ∗ 180.0 / PI

Functions

- static void * IdleModeThread (void *resource)
- int IdleFork ()
- int IdleJoin ()

Variables

- pthread_t **idle_thread**
- ThreadResource **resource**
- static int **error**

4.9.1 Detailed Description

Idle Mode Implementation.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.9.2 Function Documentation

4.9.2.1 IdleFork()

```
int IdleFork ()
```

Executes Idle Mode (concurrently), so we see how badly we messed up our code/sensors

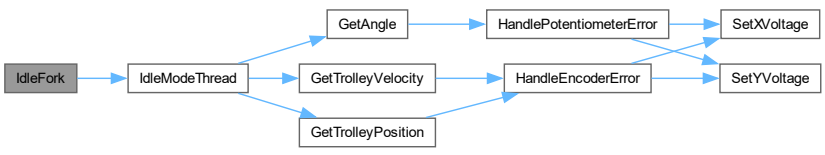
Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.9.2.2 IdleJoin()

```
int IdleJoin ()
```

Stops Idle Mode (concurrent process) and our pain

Returns

0 upon success, negative if error

4.9.2.3 IdleModeThread()

```
static void * IdleModeThread (
            void * resource)  [static]
```

The Thread Function for Idle Mode

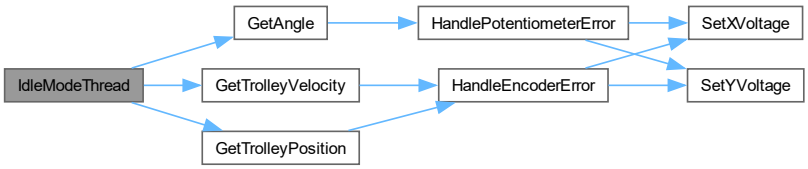
Parameters

resource	A pointer to a Resource sturcture for Idle Mode

Returns

NULL

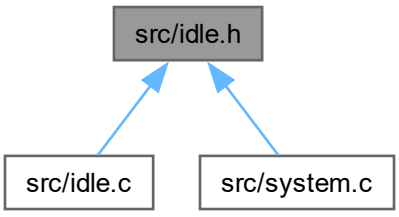
Here is the call graph for this function:



4.10 src/idle.h File Reference

Idle Mode Header.

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



Functions

- int IdleFork ()
- int IdleJoin ()

4.10.1 Detailed Description

Idle Mode Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.10.2 Function Documentation

4.10.2.1 IdleFork()

```
int IdleFork ()
```

Executes Idle Mode (concurrently), so we see how badly we messed up our code/sensors

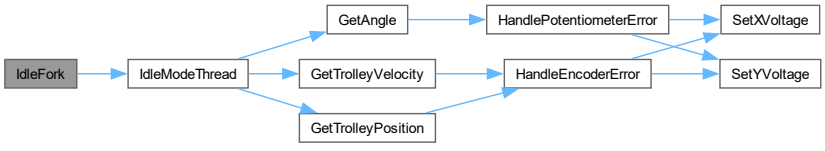
Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.10.2.2 IdleJoin()

```
int IdleJoin ()
```

Stops Idle Mode (concurrent process) and our pain

Returns

0 upon success, negative if error

4.11 idle.h

Go to the documentation of this file.

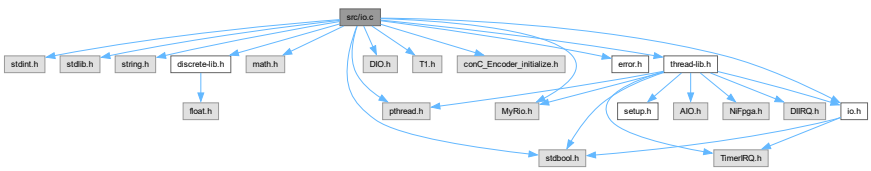
```
00001
00013 #ifndef IDLE_H_
00014 #define IDLE_H_
00015
00024 int IdleFork();
00025
00032 int IdleJoin();
00033
00034 #endif  // IDLE_H_
```

4.12 src/io.c File Reference

Sensor/Actuator (Input/Output) Interfacing Library.

```
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <math.h>
#include <pthread.h>
#include "MyRio.h"
#include "DIO.h"
#include "T1.h"
#include "conC_Encoder_initialize.h"
#include "discrete-lib.h"
#include "error.h"
#include "thread-lib.h"
#include "io.h"
```

Include dependency graph for io.c:



Macros

- #define **X_CONNECTOR_ID** 0
- #define **Y_CONNECTOR_ID** 1
- #define **POTENTIOMETER_SLOPE** -2.11 ∗ PI / 180.0
- #define **POT_V_LIM_LO** -20.0
- #define **POT_V_LIM_HI** 20.0
- #define **ENC_CNT_REV** 2000.0
- #define **M_PER_REV** 0.01267 ∗ PI
- #define ENC_2_POS(value) (value) / ENC_CNT_REV ∗ M_PER_REV
- #define ENC_2_VEL(value) (value) / (BTI_S ∗ ENC_CNT_REV) ∗ M_PER_REV
- #define **X_LIM_LO** 0.0
- #define **Y_LIM_LO** 0.0
- #define **X_LIM_HI** 0.35
- #define **Y_LIM_HI** 0.35
- #define **VEL_LIM_ABS** 1.0
- #define **CHANNELS** 16
- #define **LCD_KEYPAD_LEN** 4
- #define **UNIT_VEL** 0.15
- #define **DEL_ROW** 7
- #define **DEL_COL** 3
- #define **WAIT_CONST** 417000

Typedefs

- typedef bool Keymap[9]

Functions

- static void ∗ KeymapThread (void ∗resource)
- static int HandleEncoderError (Positions ∗curr_pos, Velocities ∗curr_vel)
- static int HandlePotentiometerError (Angles ∗curr_ang)
- static void wait ()
- int IOSetup ()
- int IOShutdown ()
- void Reset ()
- int GetReferenceVelocityCommand (Velocities ∗result)
- int GetReferenceAngleCommand (Angles ∗result)
- int GetAngle (Angles ∗result)
- int GetTrolleyPosition (Positions ∗result)
- int GetTrolleyVelocity (Velocities ∗result)
- int GetUserPosition (Angles ∗angle, Positions ∗pos, Positions ∗result)
- int GetUserVelocity (Angles ∗angle, Velocities ∗vel, Velocities ∗result)
- int SetXVoltage (Voltage voltage)
- int SetYVoltage (Voltage voltage)
- bool PressedDelete ()
- int KeyboardControlFork ()
- int KeyboardControlJoin ()
- char **getkey** ()

Variables

- static bool **reset**
- static float **potentiometer_v_x_intercept**
- static float **potentiometer_v_y_intercept**
- static MyRio_Aio **x_potentiometer**
- static MyRio_Aio **y_potentiometer**
- static MyRio_Encoder **x_encoder**
- static MyRio_Encoder **y_encoder**
- static int32_t **first_enc_state** [2]
- static int32_t **prev_enc_state** [2]
- static bool **holding_vel_set**
- static bool **holding_pos_set**
- static Velocities **holding_vel**
- static Positions **holding_pos**
- static const Encoder_StatusMask enc_st_mask
- MyRio_Aio **x_motor**
- MyRio_Aio **y_motor**
- MyRio_IrqTimer **timer**
- static MyRio_Dio **channel** [CHANNELS]
- static pthread_mutex_t **keyboard**
- static Keymap **keymap**
- static pthread_t **keymap_thread**
- static ThreadResource **keymap_resource**
- static int **error**

4.12.1 Detailed Description

Sensor/Actuator (Input/Output) Interfacing Library.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.12.2 Macro Definition Documentation

4.12.2.1 ENC_2_POS

```
#define ENC_2_POS(
              value)  (value) / ENC_CNT_REV * M_PER_REV
```

Converts a BDI quantity to meters

Parameters

value	THe BDI to convert

4.12.2.2 ENC_2_VEL

```
#define ENC_2_VEL(
              value)  (value) / (BTI_S * ENC_CNT_REV) * M_PER_REV
```

Converts a BDI/BTI quantity to meters per second

Parameters

The	value to convert

4.12.3 Typedef Documentation

4.12.3.1 Keymap

```
typedef bool Keymap[9]
```

Holds booleans indicating which buttons (1 through 9) are being pressed

4.12.4 Function Documentation

4.12.4.1 GetAngle()

```
int GetAngle (
            Angles * result)
```

Obtains the angle of the harness

Parameters

result	A return parameter, which will become the angle along both directions

Returns

0 upon success, other integers if otherwise

result, which will define the angle of the harness along both lateral directions

Here is the call graph for this function:

4.12.4.2 GetReferenceAngleCommand()

```
int GetReferenceAngleCommand (
            Angles * result)
```

Obtains the user command (for tracking)

Parameters

result	A return parameter, which will become the desired angle requested by the user

Returns

0 upon success, negative otherwise

An Angles structure, which reflects the angle requested from the user

4.12.4.3 GetReferenceVelocityCommand()

```
int GetReferenceVelocityCommand (
            Velocities * result)
```

Obtains the user command (for anti-sway)

Parameters

result	A return parameter, which will become the change in position requested by the user

Returns

0 upon success, negative otherwise

A Velocities structure, which reflects the change in position requested from the user

4.12.4.4 GetTrolleyPosition()

```
int GetTrolleyPosition (
            Positions * result)
```

Obtains the Trolley Position

Parameters

result	A return parameter, which will become the position of the trolley

Returns

0 upon success, other integers if otherwise

A Positions structure, which defines the Position of the Motor in the lateral plane

Precondition

This is called precisely once every BTI

Here is the call graph for this function:

4.12.4.5 GetTrolleyVelocity()

```
int GetTrolleyVelocity (
            Velocities * result)
```

Obtains the Trolley Velocity

Parameters

result	A return parameter, which will become the velocity of the trolley

Returns

0 upon success, other integers if otherwise

A Velocities structure, which defines the velocity of the trolley in the lateral plane

Precondition

This is called precisely once every BTI

Here is the call graph for this function:

4.12.4.6 GetUserPosition()

```
int GetUserPosition (
            Angles * angle,
            Positions * pos,
            Positions * result)
```

Obtains the User Position

Parameters

angle	The rope angle
pos	The trolley position
result	A return parameter, which will become the position of the user

Returns

0 upon success, other integers if otherwise

A Positions structure, which defines the Position of the User in the lateral plane

4.12.4.7 GetUserVelocity()

```
int GetUserVelocity (
            Angles * angle,
            Velocities * vel,
            Velocities * result)
```

Obtains the User Velocity

Parameters

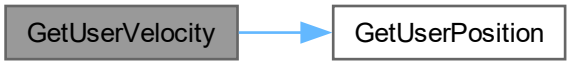
angle	The rope angle
vel	The trolley velocity
result	A return parameter, which will become the velocity of the user

Returns

0 upon success, other integers if otherwise

A Velocities structure, which defines the Velocity of the User in the lateral plane

Here is the call graph for this function:



4.12.4.8 HandleEncoderError()

```
static int HandleEncoderError (
            Positions * curr_pos,
            Velocities * curr_vel)  [inline], [static]
```

Handles Error Processing from Position/Velocity Measurements

Parameters

curr_pos	The current position
curr_vel	The current velocity

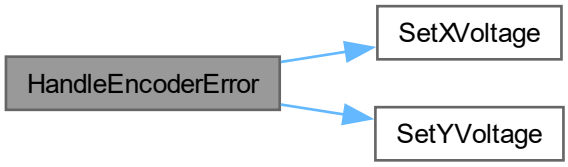
Returns

0 upon no error, negative otherwise (using the universal error codes)

Postcondition

Iff negative is returned, both motors are switched off

Here is the call graph for this function:



4.12.4.9 HandlePotentiometerError()

```
static int HandlePotentiometerError (
            Angles * curr_ang)  [inline], [static]
```

Handles Error Processing for Potentiometer Measurements

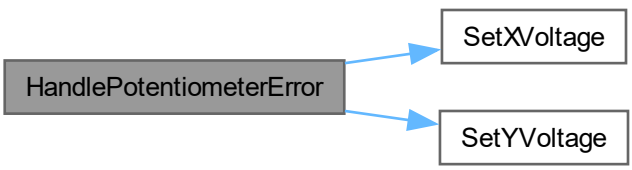
Parameters

curr_ang	The current angle reading

Returns

0 upon no error, ESTRN otherwise

Here is the call graph for this function:



4.12.4.10 IOSetup()

```
int IOSetup ()
```

Sets up the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

4.12.4.11 IOShutdown()

```
int IOShutdown ()
```

Shutsdown the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

4.12.4.12 KeyboardControlFork()

```
int KeyboardControlFork ()
```

Enables Keyboard Control for Anti-Sway (concurrently)

Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:

4.12.4.13 KeyboardControlJoin()

```
int KeyboardControlJoin ()
```

Stops Keyboard Control for Anti-Sway (concurrent process)

Returns

0 upon success, negative if error

4.12.4.14 KeymapThread()

```
static void * KeymapThread (
            void * resource)  [inline], [static]
```

Obtains the numerical buttons pressed (1 through 9)

Parameters

keymap	The thread resource to signal this thread when to stop

Returns

NULL

Postcondition

Updates keymap with all the number buttons, excluding 0, that are pressed

4.12.4.15 PressedDelete()

```
bool PressedDelete ()
```

Detects if the DEL key is pressed on the keyboard

Returns

true iff DEL is pressed on the keyboard

4.12.4.16 Reset()

```
void Reset ()
```

Resets GetTrolleyPosition and GetTrolleyVelocity by setting the velocity to zero

Postcondition

The next time GetTrolleyVelocity is called, both velocities are zero

4.12.4.17 SetXVoltage()

```
int SetXVoltage (
            Voltage voltage)
```

Sets the voltage of the X motor

Returns

0 upon success, other integers if otherwise

4.12.4.18 SetYVoltage()

```
int SetYVoltage (
            Voltage voltage)
```

Sets the voltage of the Y motor

Returns

0 upon success, other integers if otherwise

4.12.4.19 wait()

```
static void wait ()  [inline], [static]
```

Waits for approximate 5 ms

Postcondition

About 5 ms have passed

4.12.5 Variable Documentation

4.12.5.1 enc_st_mask

```
const Encoder_StatusMask enc_st_mask  [static]
```

Initial value:

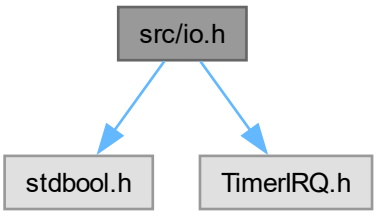
```
=
    (Encoder_StError)
```

4.13 src/io.h File Reference

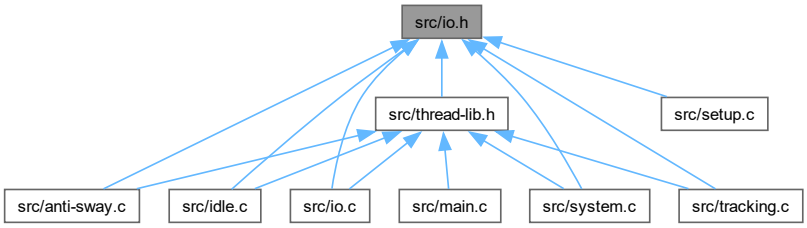
Sensor/Actuator (Input/Output) Interfacing Library Header.

```
#include <stdbool.h>
#include "TimerIRQ.h"
```

Include dependency graph for io.h:



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



Data Structures

- struct Angles
  A 2D Angle.
- struct Positions
  A 2D Position.
- struct Velocities
  A 2D Velocity.

Macros

- #define **MOTOR_V_LIM_H** 10.000
- #define **MOTOR_V_LIM_L** -10.000
- #define **R** 0.0062
- #define **K_a** 0.41
- #define **K_m** 0.11
- #define FORCE_TO_VOLTAGE(force) (force) ∗ R / (K_a ∗ K_m)
- #define **VOLTAGE_TO_FORCE**(voltage) (voltage) ∗ (K_a ∗ K_m) / R

Typedefs

- typedef float **Angle**
- typedef float **Position**
- typedef float **Velocity**
- typedef float **Voltage**

Functions

- int IOSetup ()
- int IOShutdown ()
- void Reset ()
- int GetReferenceVelocityCommand (Velocities ∗result)
- int GetReferenceAngleCommand (Angles ∗result)
- int GetAngle (Angles ∗result)
- int GetTrolleyPosition (Positions ∗result)
- int GetTrolleyVelocity (Velocities ∗result)
- int GetUserPosition (Angles ∗angle, Positions ∗pos, Positions ∗result)
- int GetUserVelocity (Angles ∗angle, Velocities ∗vel, Velocities ∗result)
- int SetXVoltage (Voltage voltage)
- int SetYVoltage (Voltage voltage)
- bool PressedDelete ()
- int KeyboardControlFork ()
- int KeyboardControlJoin ()

Variables

- MyRio_IrqTimer **timer**

4.13.1 Detailed Description

Sensor/Actuator (Input/Output) Interfacing Library Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.13.2 Macro Definition Documentation

4.13.2.1 FORCE_TO_VOLTAGE

```
#define FORCE_TO_VOLTAGE(
              force)  (force) * R / (K_a * K_m)
```

Force to Voltage Conversion

Parameters

force	An int/float/double expression, which represents the force to transmit

Postcondition

Becomes the conversion between force to the voltage to output

4.13.3 Function Documentation

4.13.3.1 GetAngle()

```
int GetAngle (
             Angles * result)
```

Obtains the angle of the harness

Parameters

result	A return parameter, which will become the angle along both directions

Returns

0 upon success, other integers if otherwise

result, which will define the angle of the harness along both lateral directions

Here is the call graph for this function:

4.13.3.2 GetReferenceAngleCommand()

```
int GetReferenceAngleCommand (
            Angles * result)
```

Obtains the user command (for tracking)

Parameters

result	A return parameter, which will become the desired angle requested by the user

Returns

0 upon success, negative otherwise

An Angles structure, which reflects the angle requested from the user

4.13.3.3 GetReferenceVelocityCommand()

```
int GetReferenceVelocityCommand (
            Velocities * result)
```

Obtains the user command (for anti-sway)

Parameters

result	A return parameter, which will become the change in position requested by the user

Returns

0 upon success, negative otherwise

A Velocities structure, which reflects the change in position requested from the user

4.13.3.4 GetTrolleyPosition()

```
int GetTrolleyPosition (
            Positions * result)
```

Obtains the Trolley Position

Parameters

result	A return parameter, which will become the position of the trolley

Returns

0 upon success, other integers if otherwise

A Positions structure, which defines the Position of the Motor in the lateral plane

Precondition

This is called precisely once every BTI

Here is the call graph for this function:

4.13.3.5 GetTrolleyVelocity()

```
int GetTrolleyVelocity (
            Velocities * result)
```

Obtains the Trolley Velocity

Parameters

result	A return parameter, which will become the velocity of the trolley

Returns

0 upon success, other integers if otherwise

A Velocities structure, which defines the velocity of the trolley in the lateral plane

Precondition

This is called precisely once every BTI

Here is the call graph for this function:

4.13.3.6 GetUserPosition()

```
int GetUserPosition (
            Angles * angle,
            Positions * pos,
            Positions * result)
```

Obtains the User Position

Parameters

angle	The rope angle
pos	The trolley position
result	A return parameter, which will become the position of the user

Returns

0 upon success, other integers if otherwise

A Positions structure, which defines the Position of the User in the lateral plane

4.13.3.7 GetUserVelocity()

```
int GetUserVelocity (
            Angles * angle,
            Velocities * vel,
            Velocities * result)
```

Obtains the User Velocity

Parameters

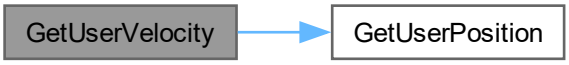
angle	The rope angle
vel	The trolley velocity
result	A return parameter, which will become the velocity of the user

Returns

0 upon success, other integers if otherwise

A Velocities structure, which defines the Velocity of the User in the lateral plane

Here is the call graph for this function:



4.13.3.8 IOSetup()

```
int IOSetup ()
```

Sets up the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

4.13.3.9 IOShutdown()

```
int IOShutdown ()
```

Shutsdown the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

4.13.3.10 KeyboardControlFork()

```
int KeyboardControlFork ()
```

Enables Keyboard Control for Anti-Sway (concurrently)

Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:

4.13.3.11 KeyboardControlJoin()

```
int KeyboardControlJoin ()
```

Stops Keyboard Control for Anti-Sway (concurrent process)

Returns

0 upon success, negative if error

4.13.3.12 PressedDelete()

```
bool PressedDelete ()
```

Detects if the DEL key is pressed on the keyboard

Returns

true iff DEL is pressed on the keyboard

4.13.3.13 Reset()

```
void Reset ()
```

Resets GetTrolleyPosition and GetTrolleyVelocity by setting the velocity to zero

Postcondition

The next time GetTrolleyVelocity is called, both velocities are zero

4.13.3.14 SetXVoltage()

```
int SetXVoltage (
            Voltage voltage)
```

Sets the voltage of the X motor

Returns

0 upon success, other integers if otherwise

4.13.3.15 SetYVoltage()

```
int SetYVoltage (
            Voltage voltage)
```

Sets the voltage of the Y motor

Returns

0 upon success, other integers if otherwise

4.14 io.h

Go to the documentation of this file.

```
00001
00013 #ifndef IO_H_
00014 #define IO_H_
00015
00016 #include <stdbool.h>
00017
00018 #include "TimerIRQ.h"
00019
00020
00021 /* Input/Output Data Types */
00022
00023 // Alias for an Angle
00024 typedef float Angle;
00025 // Alias for a Position
00026 typedef float Position;
00027 // Alias for Velocity
00028 typedef float Velocity;
00029 // Alias for Voltage
00030 typedef float Voltage;
00031
00032
00039 typedef struct {
00040     Angle x_angle;
00041     Angle y_angle;
00042 } Angles;
00043
00050 typedef struct {
00051     Position x_pos;
00052     Position y_pos;
00053 } Positions;
00054
00061 typedef struct {
00062     Velocity x_vel;
00063     Velocity y_vel;
00064 } Velocities;
00065
00066 /* Sensor Variables */
00067 // The Timer
00068 extern MyRio_IrqTimer timer;
00069
00070 /* Actuator Limits */
00071 // Motor Voltage High Limit (V)
```

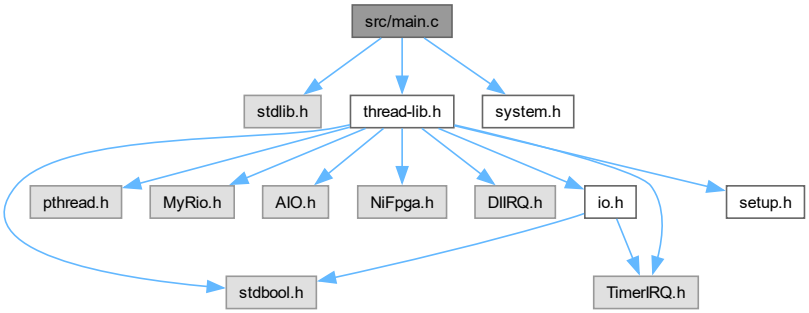
```
00072 #define MOTOR_V_LIM_H 10.000
00073 // Motor Voltage Low Limit (V)
00074 #define MOTOR_V_LIM_L -10.000
00075
00076 /* Physical Parameters */
00077 // Pulley Radius (m)
00078 #define R 0.0062
00079 // Current Constant (A/V)
00080 #define K_a 0.41
00081 // Motor Constant (Nm/A)
00082 #define K_m 0.11
00093 #define FORCE_TO_VOLTAGE(force) \
00094     (force) * R / (K_a * K_m)
00095 #define VOLTAGE_TO_FORCE(voltage) \
00096     (voltage) * (K_a * K_m) / R
00097
00098 /* Setup/Shutdown Functions */
00099
00100
00108 int IOSetup();
00109
00117 int IOShutdown();
00118
00119
00120 /* Reset Feature */
00121
00122
00131 void Reset();
00132
00133
00134 /* Sensor Functions */
00135
00136
00148 int GetReferenceVelocityCommand(Velocities *result);
00149
00161 int GetReferenceAngleCommand(Angles *result);
00162
00175 int GetAngle(Angles *result);
00176
00191 int GetTrolleyPosition(Positions *result);
00192
00207 int GetTrolleyVelocity(Velocities *result);
00208
00223 int GetUserPosition(Angles *angle, Positions *pos, Positions *result);
00224
00239 int GetUserVelocity(Angles *angle, Velocities *vel, Velocities *result);
00240
00241
00242 /* Actuator Functions */
00243
00244
00251 int SetXVoltage(Voltage voltage);
00252
00259 int SetYVoltage(Voltage voltage);
00260
00261
00262 /* Keyboard Functions */
00263
00271 bool PressedDelete();
00272
00280 int KeyboardControlFork();
00281
00287 int KeyboardControlJoin();
00288
00289 #endif  // IO_H_
```

4.15 src/main.c File Reference

Main File.

```
#include <stdlib.h>
#include "thread-lib.h"
#include "system.h"
```

Include dependency graph for main.c:



Functions

- int main (int argc, char **argv)

4.15.1 Detailed Description

Main File.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.15.2 Function Documentation

4.15.2.1 main()

```
int main (
            int argc,
            char ** argv)
```

Runs the Anti-Sway Capstone Project

Parameters

argc	Command Line Arguments (Quantity)
argv	Command Line Arguments (Contents)

Returns

0 iff success, negative otherwise

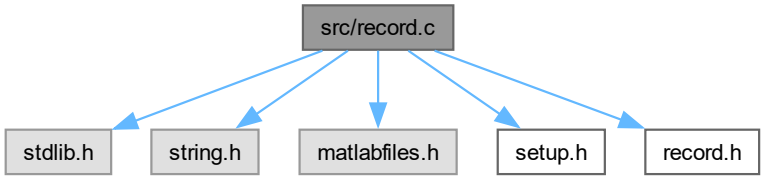
Here is the call graph for this function:

4.16 src/record.c File Reference

Data Recording Interface.

```
#include <stdlib.h>
#include <string.h>
#include "matlabfiles.h"
#include "setup.h"
#include "record.h"
```

Include dependency graph for record.c:



Data Structures

- struct DataFile_t

  Data File.

Macros

- #define **DEFAULT_NUM_FILES** 3
- #define **DEFAULT_NUM_VALS** 10
- #define **DEFAULT_RESIZE_FACTOR** 2

Functions

- static void DeallocateHelper ()
- static int ReallocateHelper (DataFile_t *f)
- FileID_t OpenDataFile (char *name, char **entry_names, int num_entries)
- int RecordData (FileID_t file, double data[ ], int data_length)
- int RecordValue (FileID_t file, char *value_name, double value)
- int SaveDataFiles ()

Variables

- static DataFile_t * **files** = NULL
- static int **num_files** = 0
- static int **capacity_files** = 0

4.16.1 Detailed Description

Data Recording Interface.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.16.2 Function Documentation

4.16.2.1 DeallocateHelper()

```
static void DeallocateHelper ()  [inline], [static]
```

Deallocates the entire module

4.16.2.2 OpenDataFile()

```
FileID_t OpenDataFile (
            char * name,
            char ** entry_names,
            int num_entries)
```

Opens a data file

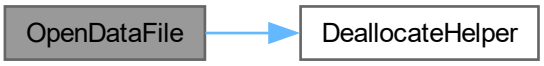
Parameters

name	The name of the file
entry_names	The name of each entry
num_entries	The number of entries

Returns

The file ID upon success, or negative upon failure

Here is the call graph for this function:



4.16.2.3 ReallocateHelper()

```
static int ReallocateHelper (
            DataFile_t * f)  [inline], [static]
```

Reallocates the entry_values for a DataFile_t

Parameters

f	A pointer to the DataFile_t to resize

Returns

0 iff success, negative upon error

Postcondition

for all 0 <= i <= f->num_entries, f->entry_values[i] is now double its capacity from before, if, at the beginning of this function, f->num_vals == f->vals->capacity

4.16.2.4 RecordData()

```
int RecordData (
            FileID_t file,
            double data[],
            int data_length)
```

Records data for each entry

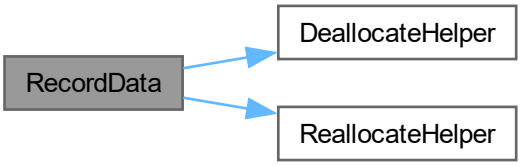
Parameters

file	The FileID_t to record upon
data	The array of data to record (in order of the entries)
data_length	The length of the data array

Returns

0 iff success, negative upon failure

Here is the call graph for this function:



4.16.2.5 RecordValue()

```
int RecordValue (
            FileID_t file,
            char * value_name,
            double value)
```

Records one-time data

Parameters

file	The FileID_t to record upon
value_name	The name of the value
value	The value to record

Returns

0 iff success, negative upon failure

4.16.2.6 SaveDataFiles()

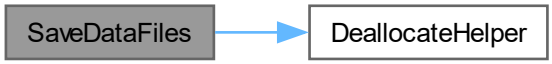
```
int SaveDataFiles ()
```

Records all data into actual files, and closes all files

Returns

0 iff success, negative upon failure

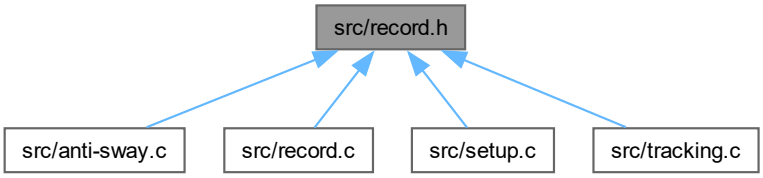
Here is the call graph for this function:



4.17 src/record.h File Reference

Data Recording Interface Header.

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



Typedefs

- typedef int **FileID_t**

Functions

- FileID_t OpenDataFile (char ∗name, char ∗∗entry_names, int num_entries)
- int RecordData (FileID_t file, double data[ ], int data_length)
- int RecordValue (FileID_t file, char ∗value_name, double value)
- int SaveDataFiles ()

4.17.1 Detailed Description

Data Recording Interface Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.17.2 Function Documentation

4.17.2.1 OpenDataFile()

```
FileID_t OpenDataFile (
            char * name,
            char ** entry_names,
            int num_entries)
```

Opens a data file

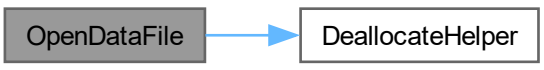
Parameters

name	The name of the file
entry_names	The name of each entry
num_entries	The number of entries

Returns

The file ID upon success, or negative upon failure

Here is the call graph for this function:



4.17.2.2 RecordData()

```
int RecordData (
            FileID_t file,
            double data[],
            int data_length)
```

Records data for each entry

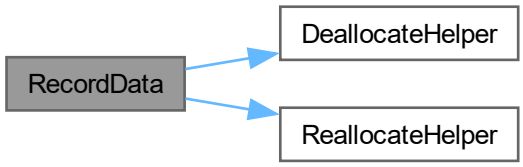
Parameters

file	The FileID_t to record upon
data	The array of data to record (in order of the entries)
data_length	The length of the data array

Returns

0 iff success, negative upon failure

Here is the call graph for this function:



4.17.2.3 RecordValue()

```
int RecordValue (
            FileID_t file,
            char * value_name,
            double value)
```

Records one-time data

Parameters

file	The FileID_t to record upon
value_name	The name of the value
value	The value to record

Returns

0 iff success, negative upon failure

4.17.2.4 SaveDataFiles()

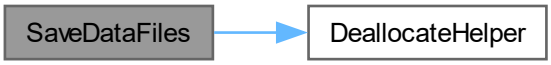
```
int SaveDataFiles ()
```

Records all data into actual files, and closes all files

Returns

0 iff success, negative upon failure

Here is the call graph for this function:



4.18 record.h

Go to the documentation of this file.

```
00001
00013 #ifndef RECORD_H_
00014 #define RECORD_H_
00015
00016 // A File
00017 typedef int FileID_t;
00018
00028 FileID_t OpenDataFile(char *name, char **entry_names, int num_entries);
00029
00039 int RecordData(FileID_t file, double data[], int data_length);
00040
00050 int RecordValue(FileID_t file, char *value_name, double value);
00051
00057 int SaveDataFiles();
00058
00059 #endif  // RECORD_H_
```

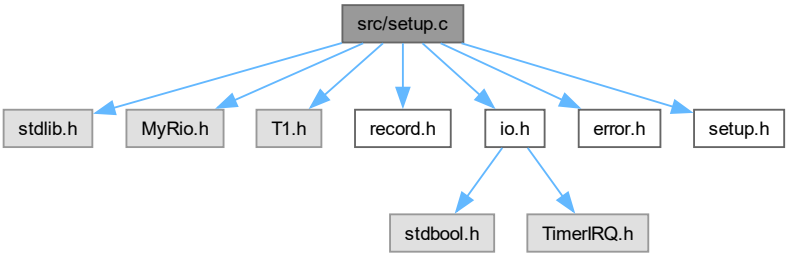
4.19 src/setup.c File Reference

System Setup.

```
#include <stdlib.h>
#include "MyRio.h"
#include "T1.h"
#include "record.h"
#include "io.h"
#include "error.h"
```

```
#include "setup.h"
```

Include dependency graph for setup.c:



Functions

- int Setup ()
- int Shutdown ()

Variables

- static int **error**
- int **u_error**

4.19.1 Detailed Description

System Setup.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.19.2 Function Documentation

4.19.2.1 Setup()

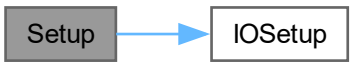
```
int Setup ()
```

Sets up the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.19.2.2 Shutdown()

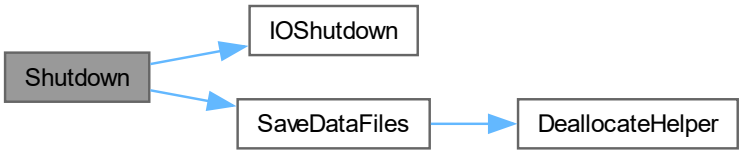
```
int Shutdown ()
```

Shuts the entire System down

Returns

0 upon success, negative otherwise

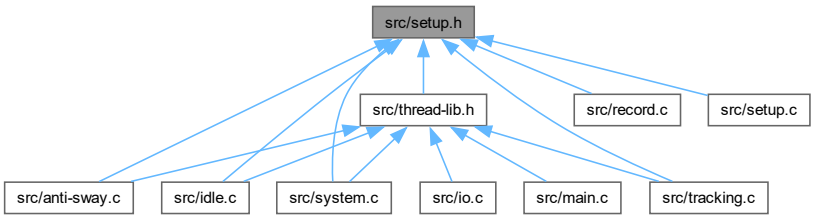
Here is the call graph for this function:



4.20 src/setup.h File Reference

System Setup Header.

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



Macros

- #define **VERIFY**(code, statement) if ((code = statement)) return code

Functions

- int Setup ()
- int Shutdown ()

4.20.1 Detailed Description

System Setup Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.20.2 Function Documentation

4.20.2.1 Setup()

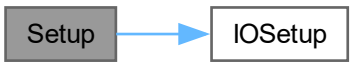
```
int Setup ()
```

Sets up the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.20.2.2 Shutdown()

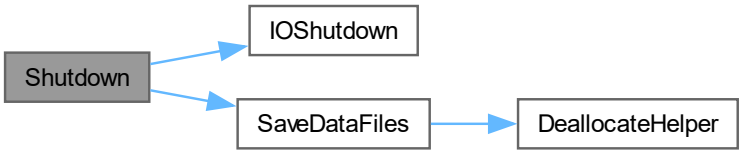
```
int Shutdown ()
```

Shuts the entire System down

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.21 setup.h

Go to the documentation of this file.

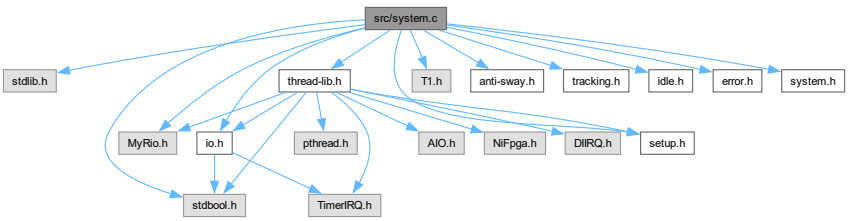
```
00001
00013 #ifndef SETUP_H_
00014 #define SETUP_H_
00015
00016
00017 /* Error Macro (for setup/shutdown) */
00018
00019
00020 #define VERIFY(code, statement) \
00021     if ((code = statement)) return code
00022
00023
00024 /* Global Setup/Shutdown Functions */
00025
00026
00033 int Setup();
00034
00035
00042 int Shutdown();
00043
00044 #endif  // SETUP_H_
```

4.22 src/system.c File Reference

System (Turing Machine)

```
#include <stdlib.h>
#include <stdbool.h>
#include "MyRio.h"
#include "T1.h"
#include "thread-lib.h"
#include "setup.h"
#include "anti-sway.h"
#include "tracking.h"
#include "idle.h"
#include "io.h"
#include "error.h"
#include "system.h"
```

Include dependency graph for system.c:



Enumerations

- enum States {
  ANTI_SWAY , **TRACKING** , **IDLE** , **MENU** ,
  ERROR , **START** , **END** }

Functions

- static int AntiSwayState ()
- static int TrackingState ()
- static int IdleState ()
- static int MenuState ()
- static int ErrorState ()
- static int StartState ()
- static int EndState ()
- int SystemExec ()

Variables

- static int(* states [ ])()
- static States **state** = START
- static int **error**

4.22.1 Detailed Description

System (Turing Machine)

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.22.2 Enumeration Type Documentation

4.22.2.1 States

```
enum States
```

The possible states

4.22.3 Function Documentation

4.22.3.1 AntiSwayState()

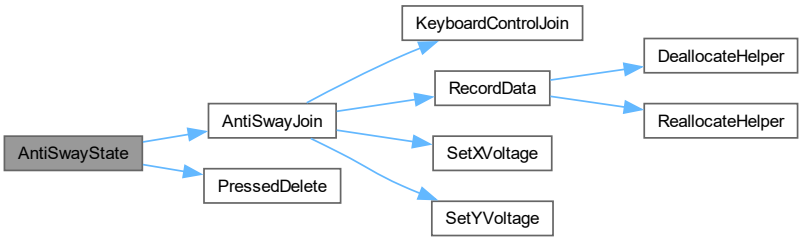
```
static int AntiSwayState ()  [static]
```

Executes the Anti-Sway State, which includes 1) Running Anti-Sway Mode 2) Executing Transitions from this State

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22.3.2 EndState()

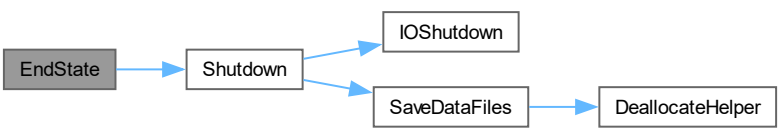
```
static int EndState ()  [static]
```

Executes the End State, which includes 1) Stopping the System 2) Stopping all Concurrent Processes 3) Deallocating all Resources

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22.3.3 ErrorState()

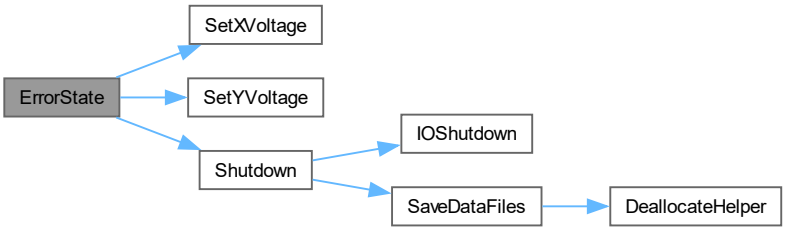
```
static int ErrorState ()  [static]
```

Executes the Error State, which includes 1) Stopping the System 2) Stopping any Concurrent Processes 3) Deallocating all Resources 4) Outputting the error

Returns

The error code from the failure

Here is the call graph for this function:



4.22.3.4 IdleState()

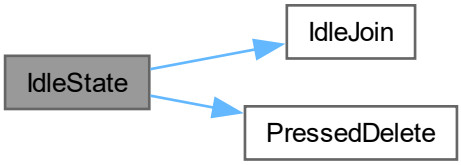
```
static int IdleState ()  [static]
```

Executes the Idle State, which includes 1) Doing Nothing 2) Executing Transitions from this State

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22.3.5 MenuState()

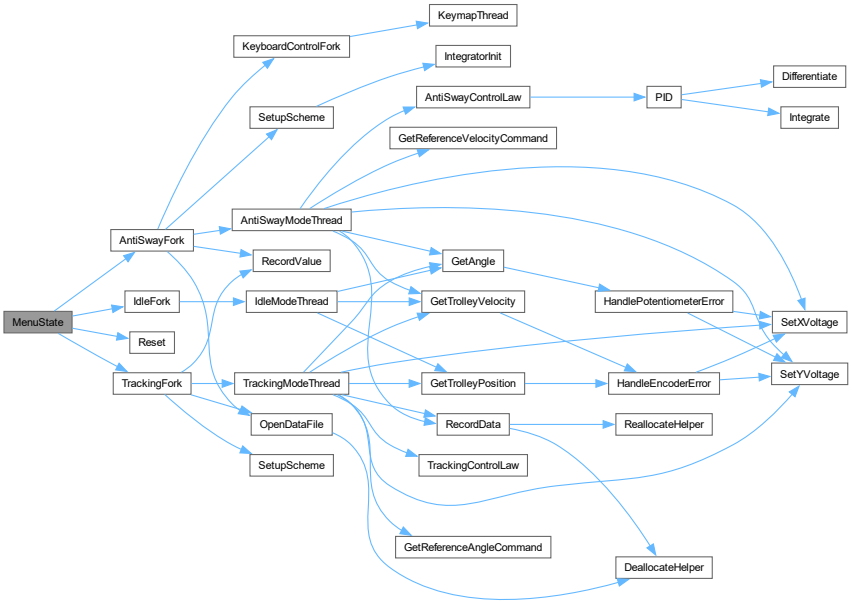
```
static int MenuState ()  [static]
```

Executes the Menu State, which includes 1) Prompting for the next state 2) Executing the next state

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22.3.6 StartState()

```
static int StartState ()  [static]
```

Executes the Start State, which includes 1) Setting up the System 2) Executing the next state Note: This is a once-only state

Returns

0 upon success, negative otherwise

Here is the call graph for this function:

4.22.3.7 SystemExec()

```
int SystemExec ()
```

Executes the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:

4.22.3.8 TrackingState()

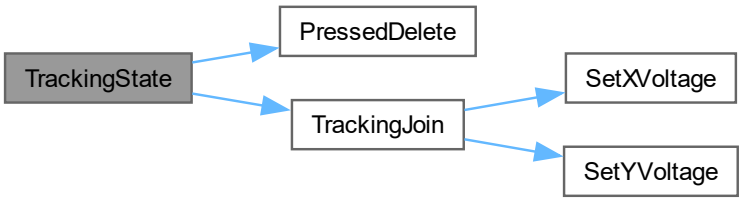
```
static int TrackingState ()  [static]
```

Executes the Tracking Mode State, which includes 1) Running Tracking Mode 2) Executing Transitions from this State

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22.4 Variable Documentation

4.22.4.1 states

```
int(* states[])() ()  [static]
```

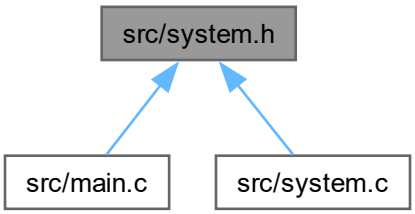
Initial value:

```
= {AntiSwayState,
                              TrackingState,
                              IdleState,
                              MenuState,
                              ErrorState,
                              StartState,
                              EndState}
```

4.23 src/system.h File Reference

System (Turing Machine) Header.

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



Functions

- int SystemExec ()

4.23.1 Detailed Description

System (Turing Machine) Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.23.2 Function Documentation

4.23.2.1 SystemExec()

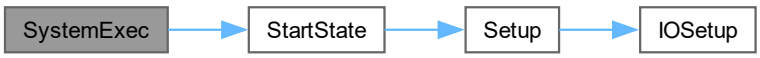
```
int SystemExec ()
```

Executes the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.24 system.h

Go to the documentation of this file.

```
00001
00013 #ifndef SYSTEM_H_
00014 #define SYSTEM_H_
00015
00016
00017 /* Execution Function */
00018
00019
00026 int SystemExec();
00027
00028 #endif  // SYSTEM_H_
```

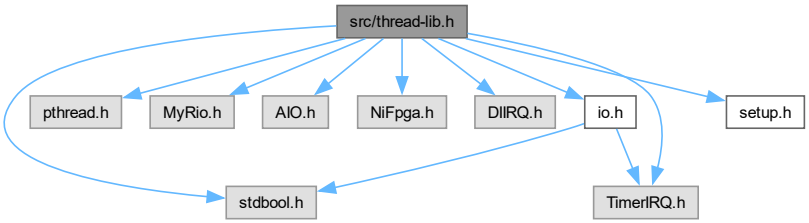
4.25 src/thread-lib.h File Reference

Thread Library.

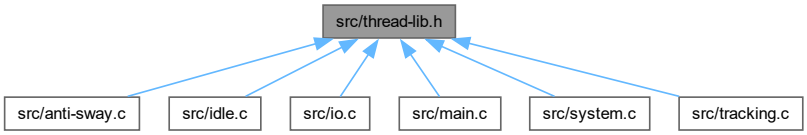
```
#include <stdbool.h>
#include <pthread.h>
#include "MyRio.h"
#include "AIO.h"
#include "NiFpga.h"
#include "DIIRQ.h"
#include "TimerIRQ.h"
#include "io.h"
```

```
#include "setup.h"
```

Include dependency graph for thread-lib.h:



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



Data Structures

- struct ThreadResource

  Parameter for Threading Functions.

Macros

- #define **BTI_US** 5000u
- #define **BTI_MS** 5u
- #define **BTI_S** 0.005
- #define **g** 9.81
- #define **PI** 3.141592653549
- #define **l** 0.47
- #define **m_dt** 2.092
- #define **m_st** 0.664
- #define **m_p** 0.765
- #define START_THREAD(thread, function, resource)
- #define REGISTER_TIMER(resource) Irq_RegisterTimerIrq(&timer, &(resource.irq_context), BTI_US)
- #define STOP_THREAD(thread, resource)
- #define UNREGISTER_TIMER(resource) Irq_UnregisterTimerIrq(&timer, resource.irq_context)
- #define TIMER_TRIGGER(irq_assert, resource)
- #define EXIT_THREAD()

Variables

- NiFpga_Session **myrio_session**

4.25.1 Detailed Description

Thread Library.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.25.2 Macro Definition Documentation

4.25.2.1 EXIT_THREAD

```
#define EXIT_THREAD()
```

Value:

```
    pthread_exit(NULL); \
    return NULL
```

Kills a Thread

Postcondition

The thread associated with the function that calls this is now gone.

4.25.2.2 REGISTER_TIMER

```
#define REGISTER_TIMER(
              resource)  Irq_RegisterTimerIrq(&timer, &(resource.irq_context), BTI_US)
```

Registers the timer (global) with a particular thread (via its resource)

Parameters

resource	The ThreadResource associated with a thread

Postcondition

The thread associated with resource is now associated with the global timer

4.25.2.3 START_THREAD

```
#define START_THREAD(
              thread,
              function,
              resource)
```

Value:

```
    resource.irq_thread_rdy = true; \
    VERIFY(error, pthread_create(&thread, NULL, function, &resource))
```

Starts a thread

Parameters

thread	The pthread_t ID variable to hold the thread's ID
function	The Thread Function to execute for the thread
resource	The ThreadResource to give the function

Precondition

An integer variable named error must be declared in this context

Postcondition

thread will contain the new PID (Process ID) of the thread

A new thread that runs function will now be running concurrently

Returns

EXIT_FAILURE upon failure to initialize the thread

4.25.2.4 STOP_THREAD

```
#define STOP_THREAD(
              thread,
              resource)
```

Value:

```
    resource.irq_thread_rdy = false; \
    VERIFY(error, pthread_join(thread, NULL))
```

Signals a Thread using a ThreadResource object to stop

Parameters

thread	The pthread_t holding the ID of the thread to stop
resource	The ThreadResource associated with the thread

Returns

EXIT_FAUILURE upon failure

Precondition

The thread uses resource, and calls EXIT_THREAD() when resource.irq_thread_rdy is set to false

Postcondition

The thread associated with pthread_t is now done

4.25.2.5 TIMER_TRIGGER

```
#define TIMER_TRIGGER(
              irq_assert,
              resource)
```

Value:

```
    Irq_Wait(resource->irq_context, \
             TIMERIRQNO, \
             &irq_assert, \
             (NiFpga_Bool *) &(resource->irq_thread_rdy)); \
    NiFpga_WriteU32(myrio_session, IRQTIMERWRITE, BTI_US); \
    NiFpga_WriteBool(myrio_session, IRQTIMERSETTIME, NiFpga_True)
```

Waits for a timer trigger (at the appropriate time step)

Parameters

irq_assert	A uint32_t that shall hold the assertion code
resource	A pointer to a ThreadResource for the thread associated with the global timer

Postcondition

irq_assert will be non-zero iff the timer has waited for the standard time step (BTI_S/MS/US)

The timer will trigger after waiting for the standard time step (BTI_S/MS/US)

4.25.2.6 UNREGISTER_TIMER

```
#define UNREGISTER_TIMER(
              resource)  Irq_UnregisterTimerIrq(&timer, resource.irq_context)
```

Dissasociates a thread with a timer (via its resource)

Parameters

resource	The ThreadResource to disassociate the global timer with

Postcondition

The thread associated with resource is now disassociated with timer

4.26 thread-lib.h

Go to the documentation of this file.

```
00001
00013 #ifndef THREAD_LIB_H_
00014 #define THREAD_LIB_H_
00015
00016 #include <stdbool.h>
00017 #include <pthread.h>
00018
00019 #include "MyRio.h"
00020 #include "AIO.h"
00021 #include "NiFpga.h"
00022 #include "DIIRQ.h"
00023 #include "TimerIRQ.h"
00024 #include "io.h"
00025
00026 #include "setup.h"
00027
00028 /* Thread Data Structures */
00029
00035 typedef struct {
00036     NiFpga_IrqContext irq_context;  // context
00037     NiFpga_Bool irq_thread_rdy;  // stop signal
00038 } ThreadResource;
00039
00040
00041 /* Time Constants */
00042
00043
00044 // The timestep, in microseconds (us)
00045 #define BTI_US 5000u
00046 // The timestep, in milliseconds (ms)
00047 #define BTI_MS 5u
00048 // The timestep, in seconds (s)
00049 #define BTI_S 0.005
00050
```

```
00051
00052 /* Physical Constants */
00053
00054
00055 // Acceleration due to Gravity (m/s^2)
00056 #define g 9.81
00057 // Pi
00058 #define PI 3.141592653549
00059 // Length of Rope (m)
00060 // TODO(nguy8tri): Define this quantity
00061 #define l 0.47
00062 // Mass of the double Trolley (kg)
00063 #define m_dt 2.092
00064 // Mass of the single Trolley (kg)
00065 #define m_st 0.664
00066 // TODO(nguy8tri): Change the masses
00067 // Mass of whole system 2.092 kg
00068 // Mass of single trolley: 0.664 kg
00069 // Mass of User 0.765 kg
00070 // Mass of User (kg)
00071 #define m_p 0.765
00072
00073
00074 /* MyRio Session */
00075 extern NiFpga_Session myrio_session;
00076
00077
00078 /* Thread Construction/Destruction */
00079
00080
00094 #define START_THREAD(thread, function, resource) \
00095     resource.irq_thread_rdy = true; \
00096     VERIFY(error, pthread_create(&thread, NULL, function, &resource))
00097
00105 #define REGISTER_TIMER(resource) \
00106     Irq_RegisterTimerIrq(&timer, &(resource.irq_context), BTI_US)
00107
00121 #define STOP_THREAD(thread, resource) \
00122     resource.irq_thread_rdy = false; \
00123     VERIFY(error, pthread_join(thread, NULL))
00124
00132 #define UNREGISTER_TIMER(resource) \
00133     Irq_UnregisterTimerIrq(&timer, resource.irq_context)
00134
00146 #define TIMER_TRIGGER(irq_assert, resource) \
00147     Irq_Wait(resource->irq_context, \
00148              TIMERIRQNO, \
00149              &irq_assert, \
00150              (NiFpga_Bool *) &(resource->irq_thread_rdy)); \
00151     NiFpga_WriteU32(myrio_session, IRQTIMERWRITE, BTI_US); \
00152     NiFpga_WriteBool(myrio_session, IRQTIMERSETTIME, NiFpga_True)
00153
00160 #define EXIT_THREAD() \
00161     pthread_exit(NULL); \
00162     return NULL
00163
00164 #endif  // THREAD_LIB_H_
```

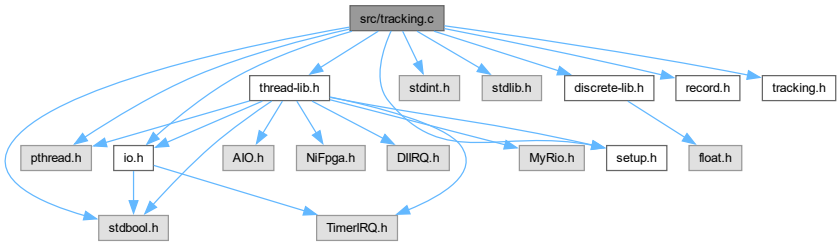
4.27 src/tracking.c File Reference

Tracking Mode Control Law.

```
#include <stdbool.h>
#include <pthread.h>
#include <stdint.h>
#include <stdlib.h>
#include "setup.h"
#include "io.h"
#include "thread-lib.h"
#include "discrete-lib.h"
#include "record.h"
```

```
#include "tracking.h"
```

Include dependency graph for tracking.c:



Data Structures

- struct TrackingControlScheme

  Tracking Mode Feedback Control Block.

Macros

- #define **NOMINAL_REFERENCE_ANGLE** 0.0
- #define **T_s** 0.1
- #define **os** 0.05
- #define **K_pi** 1.0
- #define **K_po** -2619.5
- #define **B_t** (8 ∗ m_p / T_s)
- #define **DATA_LEN** 12

Functions

- static void SetupScheme (TrackingControlScheme ∗scheme, Proportional K_o, Proportional K_i, Proportional B)
- static void ∗ TrackingModeThread (void ∗resource)
- static int TrackingControlLaw (Angle angle_ref, Angle angle_input, Velocity pos_vel, TrackingControlScheme ∗scheme, int(∗SetVoltage)(Voltage voltage))
- int TrackingFork ()
- int TrackingJoin ()

Variables

- pthread_t **tracking_thread**
- ThreadResource **resource**
- static TrackingControlScheme **x_control**
- static TrackingControlScheme **y_control**
- static int **error**
- static FileID_t **file** = -1
- static char ∗ **data_file_name** = "tracking.mat"
- static char ∗ data_names [DATA_LEN]
- static double **data** [DATA_LEN]
- static double ∗ **data_buff** = data
- static int **id** = 1

4.27.1 Detailed Description

Tracking Mode Control Law.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

4.27.2 Function Documentation

4.27.2.1 SetupScheme()

```
static void SetupScheme (
            TrackingControlScheme * scheme,
            Proportional K_o,
            Proportional K_i,
            Proportional B)  [inline], [static]
```

Sets up a TrackingControl Scheme

Parameters

scheme	The scheme to setup
K_o	The outer loop gain
K_i	The inner loop gain
B	The artificial damping to impose

Postcondition

scheme is setup with appropriate outer/inner-loop control characteristics

4.27.2.2 TrackingControlLaw()

```
static int TrackingControlLaw (
            Angle angle_ref,
            Angle angle_input,
            Velocity pos_vel,
            TrackingControlScheme * scheme,
            int(* SetVoltage )(Voltage voltage))  [inline], [static]
```

Executes 1 timestep for the Tracking Mode Control Law for its input to the plant

Parameters

angle_ref	The reference angle for Tracking Mode
angle_input	The measured rope angle for Tracking Mode
pos_vel	The measured velocity of the motor
scheme	A pointer to the TrackingControlScheme structure used to execute the control law
SetVoltage	The function that sets the voltage of the appropriate motor

Returns

0 upon success, negative otherwise

Precondition

scheme was not modified before use of this function

Postcondition

scheme is now updated with the input and outputs for the respective control scheme

4.27.2.3 TrackingFork()

```
int TrackingFork ()
```

Executes Tracking Mode (concurrently)

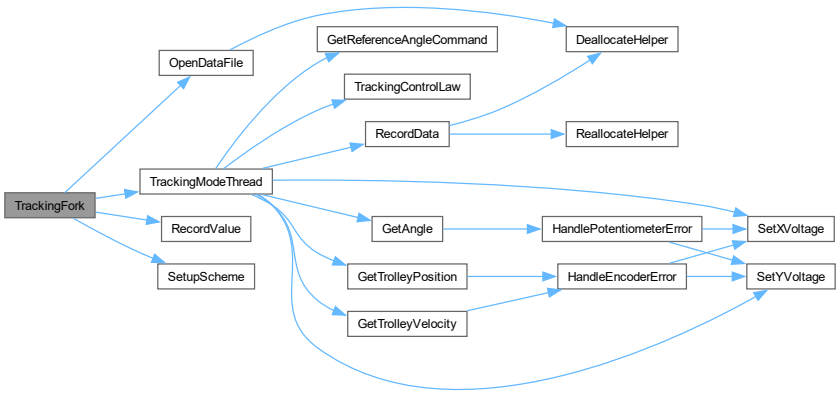
Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.27.2.4 TrackingJoin()

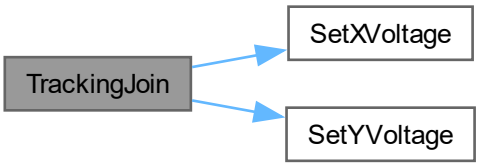
```
int TrackingJoin ()
```

Stops Tracking Mode (concurrent process)

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.27.2.5 TrackingModeThread()

```
static void * TrackingModeThread (
            void * resource)  [static]
```

The Thread Function for Tracking Mode

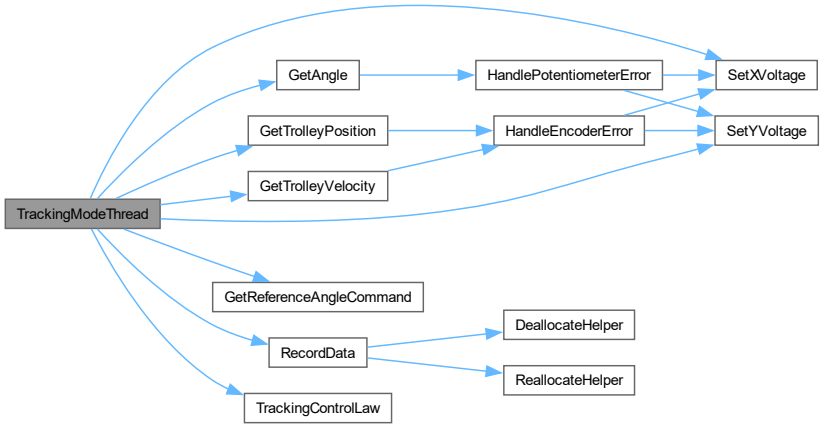
Parameters

resource	A pointer to a Resource sturcture for Tracking Mode

Returns

NULL

Here is the call graph for this function:



4.27.3 Variable Documentation

4.27.3.1 data_names

```
char* data_names[DATA_LEN]  [static]
```

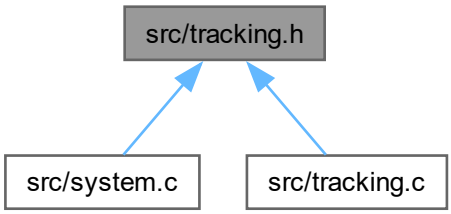
Initial value:

```
= {"id", "t",
                                    "angle_x", "angle_y",
                                    "trolley_pos_x", "trolley_pos_y",
                                    "trolley_vel_x", "trolley_vel_y",
                                    "inner_x", "voltage_x",
                                    "inner_y", "voltage_y"}
```

4.28 src/tracking.h File Reference

Tracking Mode Control Law Header.

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



Functions

- int TrackingFork ()
- int TrackingJoin ()

4.28.1 Detailed Description

Tracking Mode Control Law Header.

Author

Anti-Sway Team: Nguyen, Tri; Espinola, Malachi; Tevy, Vattanary; Hokenstad, Ethan; Neff, Callen)

Version

0.1

Date

2024-06-03

Copyright

Copyright (c) 2024

4.28.2 Function Documentation

4.28.2.1 TrackingFork()

```
int TrackingFork ()
```

Executes Tracking Mode (concurrently)

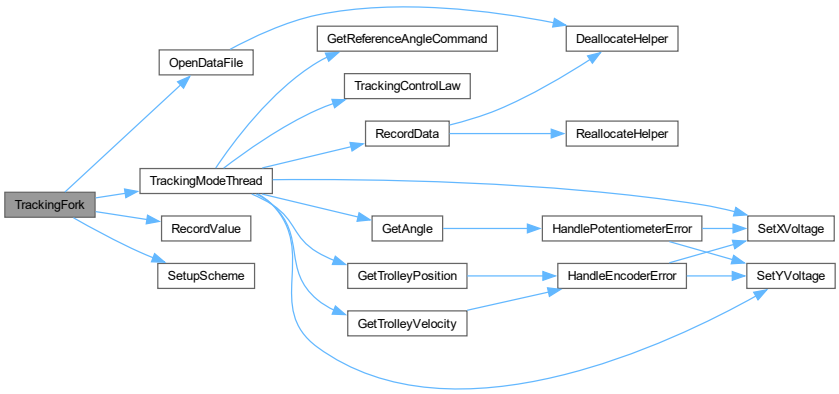
Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.28.2.2 TrackingJoin()

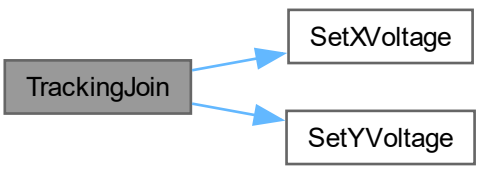
```
int TrackingJoin ()
```

Stops Tracking Mode (concurrent process)

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.29 tracking.h

Go to the documentation of this file.

```
00001
00013 #ifndef TRACKING_H_
00014 #define TRACKING_H_
00015
00016
00017 /* Execution-Dispatch Function */
00018
00019
00027 int TrackingFork();
00028
00034 int TrackingJoin();
00035
00036 #endif  // TRACKING_H_
```