Anti-Sway Capstone

1.0

Generated by Doxygen 1.11.0

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 3

2.1 File List . . . 3

3 Data Structure Documentation 5

3.1 Angles Struct Reference . . . 5

3.1.1 Detailed Description . . . 5

3.2 AntiSwayControlScheme Struct Reference . . . 5

3.2.1 Detailed Description . . . 6

3.3 Biquad Struct Reference . . . 6

3.3.1 Detailed Description . . . 6

3.4 DataFile_t Struct Reference . . . 6

3.4.1 Detailed Description . . . 7

3.5 Differentiator Struct Reference . . . 7

3.5.1 Detailed Description . . . 7

3.6 Integrator Struct Reference . . . 7

3.6.1 Detailed Description . . . 8

3.7 Positions Struct Reference . . . 8

3.7.1 Detailed Description . . . 8

3.8 ThreadResource Struct Reference . . . 8

3.8.1 Detailed Description . . . 9

3.9 TrackingControlScheme Struct Reference . . . 9

3.9.1 Detailed Description . . . 9

3.10 Velocities Struct Reference . . . 9

3.10.1 Detailed Description . . . 9

4 File Documentation 11

4.1 src/anti-sway.c File Reference . . . 11

4.1.1 Detailed Description . . . 12

4.1.2 Function Documentation . . . 13

4.1.2.1 AntiSwayControlLaw() . . . 13

4.1.2.2 AntiSwayFork() . . . 14

4.1.2.3 AntiSwayJoin() . . . 15

4.1.2.4 AntiSwayModeThread() . . . 15

4.1.2.5 SetupScheme() . . . 16

4.1.3 Variable Documentation . . . 17

4.1.3.1 data_names . . . 17

4.2 src/anti-sway.h File Reference . . . 17

4.2.1 Detailed Description . . . 18

4.2.2 Function Documentation . . . 18

4.2.2.1 AntiSwayFork() . . . 18

4.2.2.2 AntiSwayJoin() . . . 19
4.3 anti-sway.h . . . 19
4.4 src/discrete-lib.c File Reference . . . 19
4.4.1 Detailed Description . . . 20
4.4.2 Function Documentation . . . 21
4.4.2.1 Cascade() . . . 21
4.4.2.2 Differentiate() . . . 21
4.4.2.3 DifferentiatorInit() . . . 22
4.4.2.4 EvaluateBiquad() . . . 22
4.4.2.5 Integrate() . . . 23
4.4.2.6 IntegratorInit() . . . 23
4.4.2.7 PID() . . . 24
4.5 src/discrete-lib.h File Reference . . . 25
4.5.1 Detailed Description . . . 26
4.5.2 Typedef Documentation . . . 27
4.5.2.1 Proportional . . . 27
4.5.3 Function Documentation . . . 27
4.5.3.1 Cascade() . . . 27
4.5.3.2 Differentiate() . . . 28
4.5.3.3 DifferentiatorInit() . . . 28
4.5.3.4 Integrate() . . . 28
4.5.3.5 IntegratorInit() . . . 29
4.5.3.6 PID() . . . 29
4.6 discrete-lib.h . . . 30
4.7 src/error.h File Reference . . . 31
4.7.1 Detailed Description . . . 32
4.8 error.h . . . 33
4.9 src/idle.c File Reference . . . 33
4.9.1 Detailed Description . . . 34
4.9.2 Function Documentation . . . 34
4.9.2.1 IdleFork() . . . 34
4.9.2.2 IdleJoin() . . . 35
4.9.2.3 IdleModeThread() . . . 35
4.10 src/idle.h File Reference . . . 35
4.10.1 Detailed Description . . . 36
4.10.2 Function Documentation . . . 36
4.10.2.1 IdleFork() . . . 36
4.10.2.2 IdleJoin() . . . 37
4.11 idle.h . . . 37
4.12 src/io.c File Reference . . . 37
4.12.1 Detailed Description . . . 39
4.12.2 Macro Definition Documentation . . . 39

4.12.2.1 ENC_2_POS . . . 39
4.12.2.2 ENC_2_VEL . . . 40
4.12.3 Typedef Documentation . . . 40
4.12.3.1 Keymap . . . 40
4.12.4 Function Documentation . . . 40
4.12.4.1 GetAngle() . . . 40
4.12.4.2 GetReferenceAngleCommand() . . . 41
4.12.4.3 GetReferenceVelocityCommand() . . . 41
4.12.4.4 GetTrolleyPosition() . . . 41
4.12.4.5 GetTrolleyVelocity() . . . 42
4.12.4.6 GetUserPosition() . . . 43
4.12.4.7 GetUserVelocity() . . . 44
4.12.4.8 HandleEncoderError() . . . 44
4.12.4.9 HandlePotentiometerError() . . . 45
4.12.4.10 IOSetup() . . . 46
4.12.4.11 IOShutdown() . . . 46
4.12.4.12 KeyboardControlFork() . . . 46
4.12.4.13 KeyboardControlJoin() . . . 47
4.12.4.14 KeymapThread() . . . 47
4.12.4.15 PressedDelete() . . . 48
4.12.4.16 Reset() . . . 48
4.12.4.17 SetXVoltage() . . . 48
4.12.4.18 SetYVoltage() . . . 49
4.12.4.19 wait() . . . 49
4.12.5 Variable Documentation . . . 49
4.12.5.1 enc_st_mask . . . 49
4.13 src/io.h File Reference . . . 49
4.13.1 Detailed Description . . . 51
4.13.2 Macro Definition Documentation . . . 51
4.13.2.1 FORCE_TO_VOLTAGE . . . 51
4.13.3 Function Documentation . . . 51
4.13.3.1 GetAngle() . . . 51
4.13.3.2 GetReferenceAngleCommand() . . . 52
4.13.3.3 GetReferenceVelocityCommand() . . . 52
4.13.3.4 GetTrolleyPosition() . . . 53
4.13.3.5 GetTrolleyVelocity() . . . 53
4.13.3.6 GetUserPosition() . . . 54
4.13.3.7 GetUserVelocity() . . . 54
4.13.3.8 IOSetup() . . . 55
4.13.3.9 IOShutdown() . . . 55
4.13.3.10 KeyboardControlFork() . . . 55
4.13.3.11 KeyboardControlJoin() . . . 56

4.13.3.12 PressedDelete() . . . 56
4.13.3.13 Reset() . . . 56
4.13.3.14 SetXVoltage() . . . 57
4.13.3.15 SetYVoltage() . . . 57
4.14 io.h . . . 57
4.15 src/main.c File Reference . . . 58
4.15.1 Detailed Description . . . 59
4.15.2 Function Documentation . . . 59
4.15.2.1 main() . . . 59
4.16 src/record.c File Reference . . . 60
4.16.1 Detailed Description . . . 61
4.16.2 Function Documentation . . . 61
4.16.2.1 DeallocateHelper() . . . 61
4.16.2.2 OpenDataFile() . . . 61
4.16.2.3 ReallocateHelper() . . . 62
4.16.2.4 RecordData() . . . 62
4.16.2.5 RecordValue() . . . 63
4.16.2.6 SaveDataFiles() . . . 64
4.17 src/record.h File Reference . . . 64
4.17.1 Detailed Description . . . 65
4.17.2 Function Documentation . . . 65
4.17.2.1 OpenDataFile() . . . 65
4.17.2.2 RecordData() . . . 66
4.17.2.3 RecordValue() . . . 66
4.17.2.4 SaveDataFiles() . . . 67
4.18 record.h . . . 67
4.19 src/setup.c File Reference . . . 67
4.19.1 Detailed Description . . . 68
4.19.2 Function Documentation . . . 69
4.19.2.1 Setup() . . . 69
4.19.2.2 Shutdown() . . . 69
4.20 src/setup.h File Reference . . . 70
4.20.1 Detailed Description . . . 70
4.20.2 Function Documentation . . . 71
4.20.2.1 Setup() . . . 71
4.20.2.2 Shutdown() . . . 71
4.21 setup.h . . . 72
4.22 src/system.c File Reference . . . 72
4.22.1 Detailed Description . . . 73
4.22.2 Enumeration Type Documentation . . . 73
4.22.2.1 States . . . 73
4.22.3 Function Documentation . . . 74

4.22.3.1 AntiSwayState() . . . 74
4.22.3.2 EndState() . . . 74
4.22.3.3 ErrorState() . . . 75
4.22.3.4 IdleState() . . . 75
4.22.3.5 MenuState() . . . 76
4.22.3.6 StartState() . . . 76
4.22.3.7 SystemExec() . . . 77
4.22.3.8 TrackingState() . . . 77
4.22.4 Variable Documentation . . . 78
4.22.4.1 states . . . 78
4.23 src/system.h File Reference . . . 78
4.23.1 Detailed Description . . . 78
4.23.2 Function Documentation . . . 79
4.23.2.1 SystemExec() . . . 79
4.24 system.h . . . 79
4.25 src/thread-lib.h File Reference . . . 79
4.25.1 Detailed Description . . . 81
4.25.2 Macro Definition Documentation . . . 81
4.25.2.1 EXIT_THREAD . . . 81
4.25.2.2 REGISTER_TIMER . . . 81
4.25.2.3 START_THREAD . . . 81
4.25.2.4 STOP_THREAD . . . 82
4.25.2.5 TIMER_TRIGGER . . . 82
4.25.2.6 UNREGISTER_TIMER . . . 83
4.26 thread-lib.h . . . 83
4.27 src/tracking.c File Reference . . . 84
4.27.1 Detailed Description . . . 86
4.27.2 Function Documentation . . . 86
4.27.2.1 SetupScheme() . . . 86
4.27.2.2 TrackingControlLaw() . . . 86
4.27.2.3 TrackingFork() . . . 87
4.27.2.4 TrackingJoin() . . . 88
4.27.2.5 TrackingModeThread() . . . 88
4.27.3 Variable Documentation . . . 89
4.27.3.1 data_names . . . 89
4.28 src/tracking.h File Reference . . . 89
4.28.1 Detailed Description . . . 89
4.28.2 Function Documentation . . . 90
4.28.2.1 TrackingFork() . . . 90
4.28.2.2 TrackingJoin() . . . 90
4.29 tracking.h . . . 91

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Angles
: A 2D Angle 5

AntiSwayControlScheme
: Anti-Sway Mode Feedback Control Block 5

Biquad
: Biquad 6

DataFile_t
: Data File 6

Differentiator
: Control Block: Differentiator 7

Integrator
: Control Block: Integrator 7

Positions
: A 2D Position 8

ThreadResource
: Parameter for Threading Functions 8

TrackingControlScheme
: Tracking Mode Feedback Control Block 9

Velocities
: A 2D Velocity 9

Chapter 2

File Index

2.1 File List

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

C:/Users/tring/PycharmProjects/Capstone-Stuff/src/anti-sway.c
Anti-Sway Control Law Implementation . . . 11
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/anti-sway.h
Anti-Sway Control Law Header . . . 17
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/conC_Encoder_initialize.h . . . ??
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.c
Discrete Control Law Implementation Library . . . 19
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.h
Discrete Control Law Implementation Library Header . . . 25
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/error.h
Universal Error Library . . . 31
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/idle.c
Idle Mode Implementation . . . 33
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/idle.h
Idle Mode Header . . . 35
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.c
Sensor/Actuator (Input/Output) Interfacing Library . . . 37
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.h
Sensor/Actuator (Input/Output) Interfacing Library Header . . . 49
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/main.c
Main File . . . 58
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/record.c
Data Recording Interface . . . 60
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/record.h
Data Recording Interface Header . . . 64
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/setup.c
System Setup . . . 67
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/setup.h
System Setup Header . . . 70
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/system.c
System (Turing Machine) . . . 72
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/system.h
System (Turing Machine) Header . . . 78
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/thread-lib.h
Thread Library . . . 79

C:/Users/tring/PycharmProjects/Capstone-Stuff/src/tracking.c
Tracking Mode Control Law 84
C:/Users/tring/PycharmProjects/Capstone-Stuff/src/tracking.h
Tracking Mode Control Law Header 89

Chapter 3

Data Structure Documentation

3.1 Angles Struct Reference

A 2D Angle.

`#include <io.h>`

Data Fields

- Angle **x_angle**
  ! Angle Parallel to X Direction
- Angle **y_angle**
  ! Angle Parallel to Y Direction

3.1.1 Detailed Description

A 2D Angle.

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

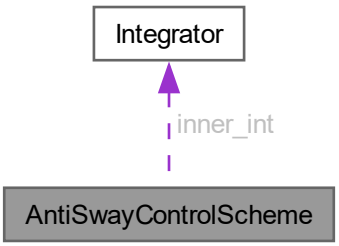
The documentation for this struct was generated from the following file:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.h

3.2 AntiSwayControlScheme Struct Reference

Anti-Sway Mode Feedback Control Block.

Collaboration diagram for AntiSwayControlScheme:



Data Fields

- Proportional **outer_feedback**
  Outer feedback.
- Proportional **inner_prop**
  Inner PI Proportional Gain.
- Integrator **inner_int**
  Innner PI Integral Term.

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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/anti-sway.c

3.3 Biquad Struct Reference

Biquad.

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

Data Fields

- double numerator [3]

  The numerator coefficients, in decreasing order of time delays.
- double denominator [3]

  The denominator coefficients, in decreasing order of time delays.
- double prev_input [2]

  The previous inputs, in increasing time delays.
- double prev_output [2]

  The previous outputs, in increasing time delays.

3.3.1 Detailed Description

Biquad.

A struct representing a biquad

3.3.2 Field Documentation

3.3.2.1 denominator

```
double denominator[3]
```

The denominator coefficients, in decreasing order of time delays.

The denominator coefficients, in decreasing order of time delays (z^0, z^-1, z^-2)

3.3.2.2 numerator

```
double numerator[3]
```

The numerator coefficients, in decreasing order of time delays.

The numerator coefficients, in decreasing order of time delays (z^0, z^-1, z^-2)

3.3.2.3 prev_input

```
double prev_input[2]
```

The previous inputs, in increasing time delays.

The previous inputs, in increasing time delays (z^-1, z^-2)

3.3.2.4 prev_output

```
double prev_output[2]
```

The previous outputs, in increasing time delays.

The previous outputs, in increasing time delays (z^-1, z^-2)

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

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.h

3.4 DataFile_t Struct Reference

Data File.

Data Fields

- MATFILE * **file**
  The MATFILE that this DataFile will store its data into.
- int **num_entries**
  The number of arrays in this file.
- char ** **entry_names**
  The names of all arrays in this file.
- int **num_vals**
  The number of values in each array.
- int **vals_capacity**
  The capacity of the arrays in the data structure below.
- double ** **entry_values**
  A pointer to pointers to arrays for the data being stored (2D array)

3.4.1 Detailed Description

Data File.

Internal Representation of a Data File

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

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/record.c

3.5 Differentiator Struct Reference

Control Block: Differentiator.

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

Data Fields

- Proportional **gain**
  Differential Gain (with Timestep)
- double **prev_input**
  Previous input.
- double **prev_output**
  Previous 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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.h

3.6 Integrator Struct Reference

Control Block: Integrator.

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

Data Fields

- Proportional **gain**
  Integral Gain (with Timestep)
- double **prev_input**
  Previous input.
- double **prev_output**
  Previous 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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.h

3.7 Positions Struct Reference

A 2D Position.

`#include <io.h>`

Data Fields

- Position **x_pos**
  ! X Position
- Position **y_pos**
  ! Y Position

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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.h

3.8 ThreadResource Struct Reference

Parameter for Threading Functions.

`#include <thread-lib.h>`

Data Fields

- NiFpga_IrqContext **irq_context**
  context
- NiFpga_Bool **irq_thread_rdy**
  stop signal

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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/thread-lib.h

3.9 TrackingControlScheme Struct Reference

Tracking Mode Feedback Control Block.

Data Fields

- Proportional **combined_constants**
  ! Combined Outer-Loop Constant
- Proportional **damping**
  ! Artifical Damping (Inner Loop Feedback Gain)

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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/tracking.c

3.10 Velocities Struct Reference

A 2D Velocity.

`#include <io.h>`

Data Fields

- Velocity **x_vel**
  ! X Velocity
- Velocity **y_vel**
  ! Y Velocity

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:

- C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.h

Chapter 4

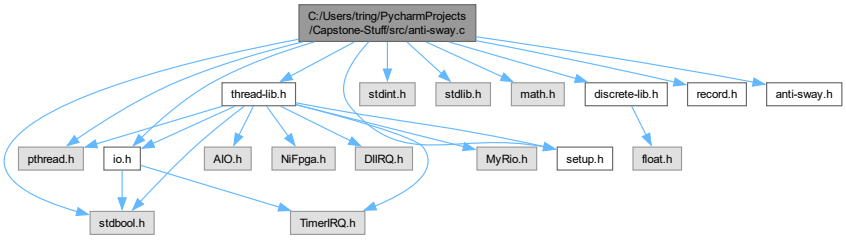
File Documentation

4.1 C:/Users/tring/PycharmProjects/Capstone-Stuff/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

  The number of entries.
- #define **TUNING**

  Tuning Mode.
- #define **TUNING_DATA_LEN** 10

  The number of array entries within the tuning file.
- #define **LR_X** 250

  Learning Rate in X direction.
- #define **LR_Y** 250

  Learning Rate in Y direction.
- #define ZERO_GRAD()

Functions

- static void SetupScheme (AntiSwayControlScheme *scheme, Proportional K_p, Proportional K_i, Proportional m)

  Sets up the Anti-Sway Control Law (its feedback path)
- static void * AntiSwayModeThread (void *resource)

  Runs Anti-Sway.
- static int AntiSwayControlLaw (Velocity vel_ref, Angle angle_input, Velocity vel_input, AntiSwayControlScheme *scheme, int(*SetVoltage)(Voltage voltage))

  Executes an iteration of the feedback path for Anti-Sway.
- int AntiSwayFork ()

  Executes Anti-Sway Mode.
- int AntiSwayJoin ()

  Stops Anti-Sway Mode.

Variables

- pthread_t **anti_sway_thread** = NULL

  Thread ID.
- ThreadResource **anti_sway_resource**

  Thread Resources (Shared Resources)
- static double **K_ptx** = 51.55550206284189

  The proportional constant for inner-loop.
- static double **K_itx** = 33.28586146285062

  The integral constant for inner-loop control.
- static double **K_pty** = 0.8*55.65965893434064

  The proportional constant for inner-loop.
- static double **K_ity** = 0.8*31.59324977878787

  The integral constant for inner-loop control.
- static AntiSwayControlScheme **x_control**

  The Control Scheme for the X Motor.
- static AntiSwayControlScheme **y_control**

  The Control Scheme for the Y Motor.
- static int **error**

  Local Error Code.
- static FileID_t **file** = -1

The file ID.

- static char * **data_file_name** = "anti-sway.mat"

  The file Name.
- static char * data_names [DATA_LEN]

  The data names.
- static double **data** [DATA_LEN]

  Buffer for data.
- static double * **data_buff** = data

  Pointer to next data point to insert into buffer.
- static int **id** = 1

  ID variable.
- static double **t** = 0.0

  timestamp
- static FileID_t **tuning_file** = -1

  The tuning file.
- static char * **tuning_file_name** = "anti-sway-tuning.mat"

  The name of the tuning file.
- static char * tuning_data_names [TUNING_DATA_LEN]

  The names of the array entries for the tuning file.
- static double dKp [2]
- static double dKi [2]
- static int total_pts [2]
- static double prev_int_Kp [2][550]
- static double prev_int_Ki [2][550]
- static int prev_int_i = 0
- static bool **int_Kp_first** = true

  Indicates (false) if prev_int_Kp has any valid data in it.
- static bool **int_Ki_first** = true

  Indicates (false) if prev_int_Ki has any valid data in it.
- static double **prev_Kp** [2]

  Stores previous Kp values in both x and y directions.
- static double **prev_Ki** [2]

  Stores previous Ki value sin both x and y directions.

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 Macro Definition Documentation

4.1.2.1 ZERO_GRAD

```
#define ZERO_GRAD()
```

Value:

```
    dKp[0] = 0.0; \
    dKp[1] = 0.0; \
    dKi[0] = 0.0; \
    dKi[1] = 0.0; \
    total_pts[0] = 0; \
    total_pts[1] = 0;
```

Zeros out Gradients

Postcondition

Zeros out dKp and dKi

4.1.3 Function Documentation

4.1.3.1 AntiSwayControlLaw()

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

Executes an iteration of the feedback path for Anti-Sway.

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

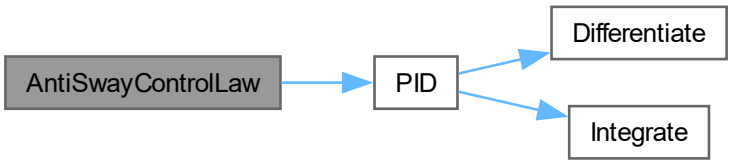
Precondition

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:



4.1.3.2 AntiSwayFork()

```
int AntiSwayFork ()
```

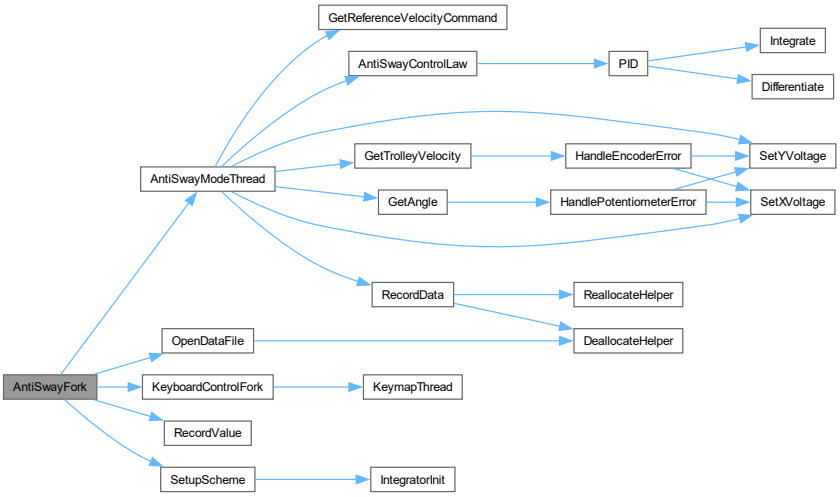
Executes Anti-Sway Mode.

Executes Anti-Sway Mode (concurrently)

Precondition

Anti-Sway Mode is not already running

Here is the call graph for this function:

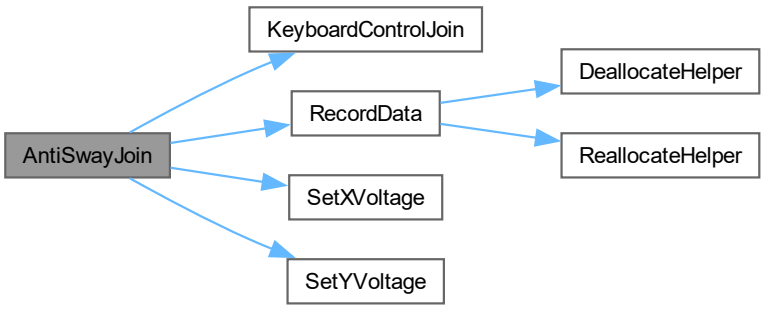


4.1.3.3 AntiSwayJoin()

```
int AntiSwayJoin ()
```

Stops Anti-Sway Mode.

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



4.1.3.4 AntiSwayModeThread()

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

Runs Anti-Sway.

The Thread Function for Anti-Sway Mode

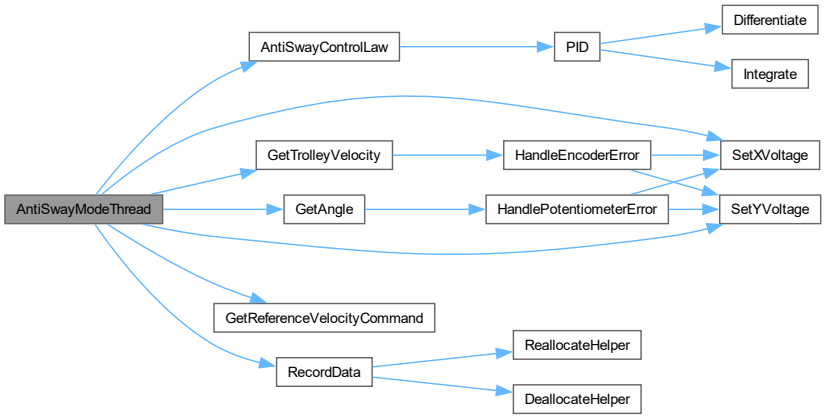
Parameters

resource	A pointer to a Resource sturcture for Tracking Mode

Returns

NULL

Here is the call graph for this function:



4.1.3.5 SetupScheme()

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

Sets up the Anti-Sway Control Law (its feedback path)

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:

4.1.4 Variable Documentation

4.1.4.1 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"}
```

The data names.

4.1.4.2 dKi

```
double dKi[2]  [static]
```

The gradient component of the loss with respect to Ki, for both x and y directions

4.1.4.3 dKp

```
double dKp[2]  [static]
```

The gradient component of the loss with respect to Kp, for both x and y directions

4.1.4.4 prev_int_i

```
int prev_int_i = 0  [static]
```

The counter that tells the program where we are along a column within above 2 arrays

4.1.4.5 prev_int_Ki

```
double prev_int_Ki[2][550]  [static]
```

Previous integral outputs with respect to change in Ki, for both x and y directions

4.1.4.6 prev_int_Kp

```
double prev_int_Kp[2][550]  [static]
```

Previous integral outputs with respect to change in Kp, for both x and y directions

4.1.4.7 total_pts

```
int total_pts[2]  [static]
```

The total number of data points used for dKp and dKi, for both x and y directions

4.1.4.8 tuning_data_names

```
char* tuning_data_names[TUNING_DATA_LEN]  [static]
```

Initial value:

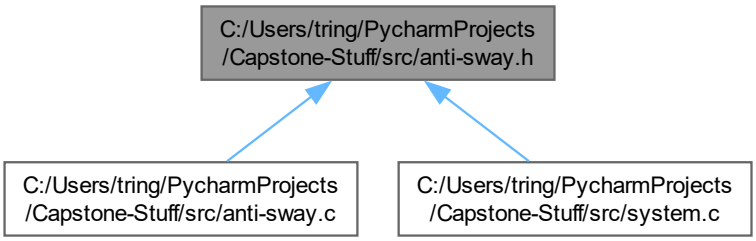
```
= {"count_x", "count_y",
                                                    "dKp_x", "dKi_x", "dKp_y", "dKi_y",
                                                    "Kp_x", "Ki_x", "Kp_y", "Ki_y"}
```

The names of the array entries for the tuning file.

4.2 C:/Users/tring/PycharmProjects/Capstone-Stuff/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 ()

  Executes Anti-Sway Mode.
- int AntiSwayJoin ()

  Stops Anti-Sway Mode.

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

Copyright (c) 2024

4.2.2 Function Documentation

4.2.2.1 AntiSwayFork()

```
int AntiSwayFork ()
```

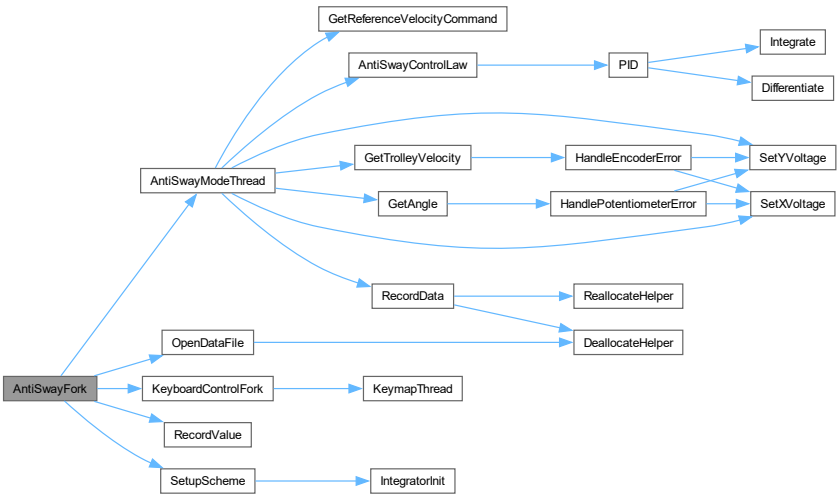
Executes Anti-Sway Mode.

Executes Anti-Sway Mode (concurrently)

Precondition

Anti-Sway Mode is not already running

Here is the call graph for this function:

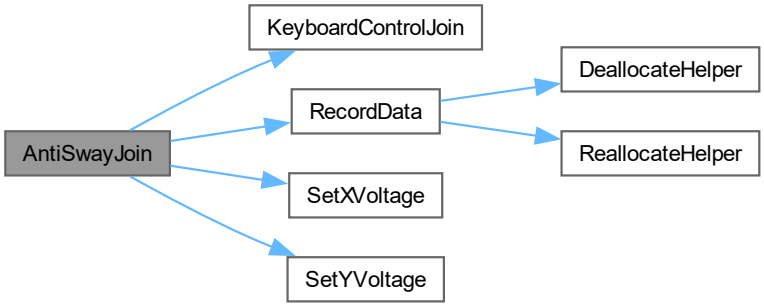


4.2.2.2 AntiSwayJoin()

```
int AntiSwayJoin ()
```

Stops Anti-Sway Mode.

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
00026 int AntiSwayFork();
00027
00033 int AntiSwayJoin();
00034
00035 #endif  // ANTI_SWAY_H_
```

4.4 conC_Encoder_initialize.h

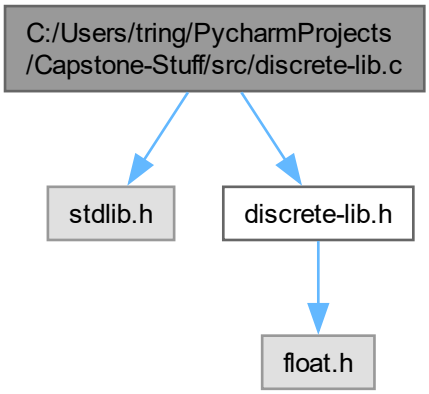
```
00001 //
00002 //  conC_Encoder_initialize.h
00003 //
00004 //
00005 //  Created by JOSEPH L GARBINI on 12/28/17.
00006 //
00007
00008 #ifndef conC_Encoder_initialize_h
00009 #define conC_Encoder_initialize_h
00010
00011 #include <stdio.h>
00012 #include "MyRio.h"
00013 #include "Encoder.h"
00014
00015 NiFpga_Status    conC_Encoder_initialize(NiFpga_Session myrio_session, MyRio_Encoder *encCp, int iE);
00016
00017 #endif /* conC_Encoder_initialize_h */
```

4.5 C:/Users/tring/PycharmProjects/Capstone-Stuff/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)

  Saturates a value.

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)

4.5.1 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

4.5.2 Macro Definition Documentation

4.5.2.1 SATURATE

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

Saturates a value.

Saturates a value to be between some low and high value

Parameters

val	The numerical value to saturate
lo	The numerical lower limit
hi	The numerical upper limit

Evaluates to val iff lo <= val <= hi, lo iff val < lo and hi iff val > hi

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

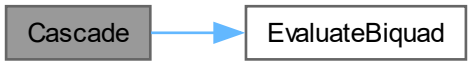
Precondition

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 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

4.5.3.5 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

4.5.3.6 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

4.5.3.7 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 all non-NULL Control Blocks

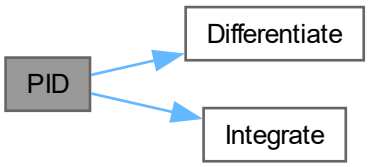
If p, i or d is NULL, then those NULL terms don't contribute

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

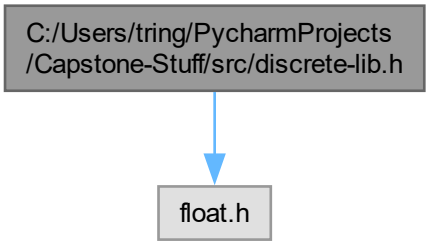
Here is the call graph for this function:



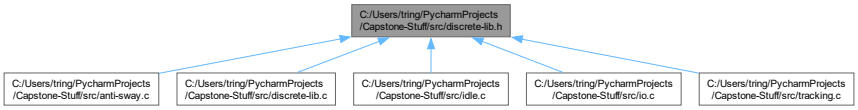
4.6 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/discrete-lib.h File Reference

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

  Positive Infinity.
- #define **NEG_INF** (-DBL_MAX)

  Negative Infinity.

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)

4.6.1 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

Copyright (c) 2024

4.6.2 Typedef Documentation

4.6.2.1 Proportional

```
typedef float Proportional
```

Control Block: Proportion.

A proportional constant

4.6.3 Function Documentation

4.6.3.1 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

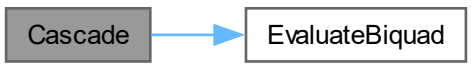
Precondition

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:



4.6.3.2 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

4.6.3.3 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

4.6.3.4 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

4.6.3.5 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

4.6.3.6 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 all non-NULL Control Blocks

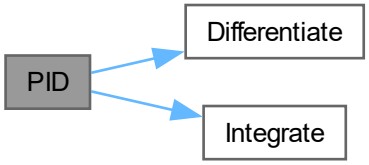
If p, i or d is NULL, then those NULL terms don't contribute

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:



4.7 discrete-lib.h

Go to the documentation of this file.

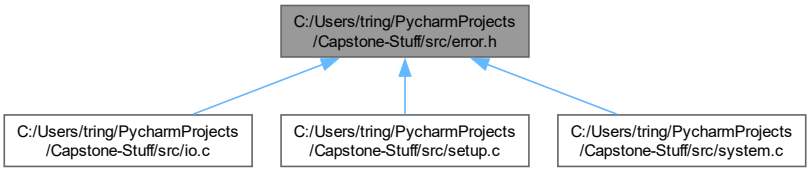
```
00001
00013 #ifndef DISCRETE_LIB_H_
00014 #define DISCRETE_LIB_H_
00015
00016 #include <float.h>
00017
00018
00019 /* Non-saturation constants */
00020
00021
00023 #define POS_INF DBL_MAX
00025 #define NEG_INF (-DBL_MAX)
00026
00027
00028 /* Discrete-Time Data Structures */
00029
00030
00036 typedef struct {
00038
00040     double numerator[3];
00042
00044     double denominator[3];
00046
00048     double prev_input[2];
00050
00052     double prev_output[2];
00053 } Biquad;
00054
00060 typedef float Proportional;
00061
00067 typedef struct {
00068     Proportional gain;
00069     double prev_input;
00070     double prev_output;
00071 } Integrator;
00072
00078 typedef struct {
00079     Proportional gain;
00080     double prev_input;
00081     double prev_output;
00082 } Differentiator;
00083
00084
00085 /* Initialization Functions */
00086
00087
00099 void IntegratorInit(Proportional gain, double timestep, Integrator *result);
00100
00112 void DifferentiatorInit(Proportional gain,
00113                         double timestep,
```

```
00114                       Differentiator *result);
00115
00116
00117 /* Time-Stepping Functions */
00118
00119
00136 inline double Cascade(double input,
00137                       Biquad sys[],
00138                       int size,
00139                       double lower_lim,
00140                       double upper_lim);
00141
00156 inline double Integrate(double input,
00157                         Integrator *term,
00158                         double lower_lim,
00159                         double upper_lim);
00160
00175 inline double Differentiate(double input,
00176                             Differentiator *term,
00177                             double lower_lim,
00178                             double upper_lim);
00179
00198 inline double PID(double input,
00199                   Proportional *p,
00200                   Integrator *i,
00201                   Differentiator *d,
00202                   double lower_lim,
00203                   double upper_lim);
00204
00205 #endif  // DISCRETE_LIB_H_
```

4.8 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/error.h File Reference

Universal Error Library.

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



Macros

- #define **ENKWN** -1

  Unknown Exception.
- #define **EOTBD** -2

  Out of Bounds Error.
- #define **EVTYE** -3

  Velocity Exceeded Error.
- #define **ESTRN** -4

  Angle Sensor Saturation Error.
- #define **EENCR** -5

  Encoder Error.

Variables

- int u_error

  Ther universal error code.

4.8.1 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

Copyright (c) 2024

4.8.2 Variable Documentation

4.8.2.1 u_error

```
int u_error  [extern]
```

Ther universal error code.

Ther universal error code.

4.9 error.h

Go to the documentation of this file.

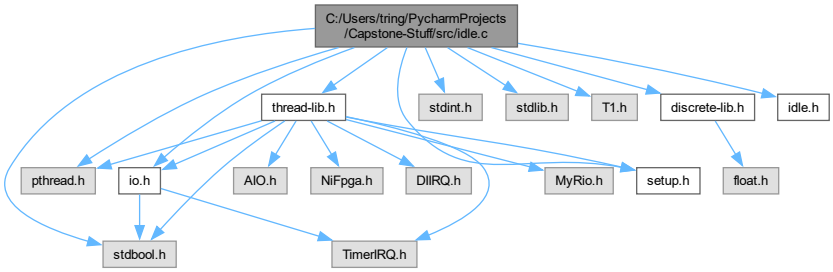
```
00001
00013 #ifndef ERROR_H_
00014 #define ERROR_H_
00015
00016 /* Universal Error Codes */
00017
00018
00019 /* Error Macro */
00021 extern int u_error;
00022
00023
00024 /* I/O Error Codes */
00025
00027 #define ENKWN -1
00029 #define EOTBD -2
00031 #define EVTYE -3
00033 #define ESTRN -4
00035 #define EENCR -5
00036
00037
00038 #endif  // ERROR_H_
```

4.10 C:/Users/tring/PycharmProjects/Capstone-Stuff/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)

  Idle Mode Thread Function.
- int IdleFork ()
- int IdleJoin ()

Variables

- pthread_t **idle_thread**

  Thread ID.
- ThreadResource **resource**

  Thread Resources (Shared Resources)
- static int **error**

  Local Error Code.

4.10.1 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

4.10.2 Function Documentation

4.10.2.1 IdleFork()

```
int IdleFork ()
```

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

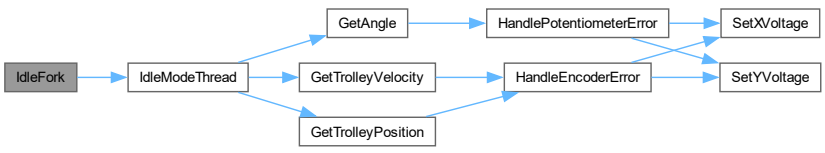
Postcondition

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

4.10.2.3 IdleModeThread()

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

Idle Mode Thread Function.

The Thread Function for Idle Mode

Parameters

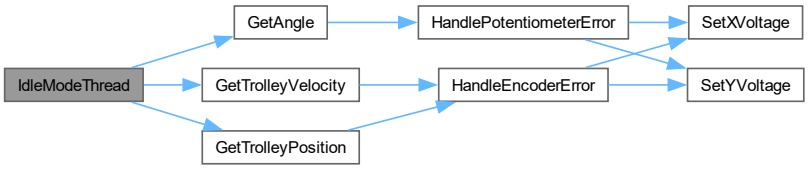
resource	A pointer to a Resource sturcture for Idle Mode

Returns

NULL

How many decimal places to include

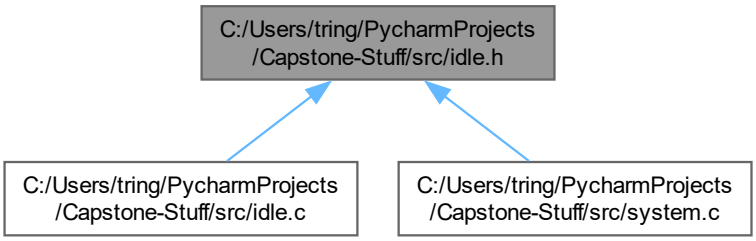
Radians to Degrees Conversion FactorHere is the call graph for this function:



4.11 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/idle.h File Reference

Idle Mode Header.

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



Functions

- int IdleFork ()
- int IdleJoin ()

4.11.1 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

4.11.2 Function Documentation

4.11.2.1 IdleFork()

```
int IdleFork ()
```

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

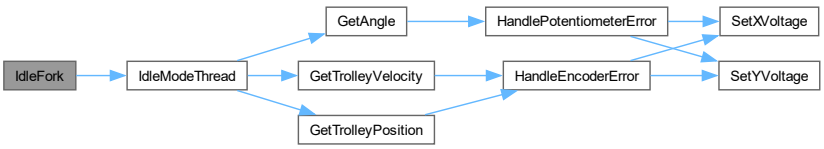
Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.11.2.2 IdleJoin()

```
int IdleJoin ()
```

Stops Idle Mode (concurrent process) and our pain

Returns

0 upon success, negative if error

4.12 idle.h

Go to the documentation of this file.

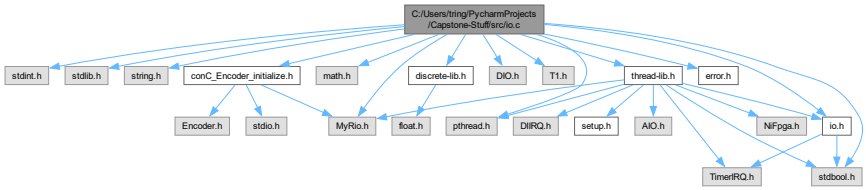
```
00001
00013 #ifndef IDLE_H_
00014 #define IDLE_H_
00015
00024 int IdleFork();
00025
00032 int IdleJoin();
00033
00034 #endif  // IDLE_H_
```

4.13 C:/Users/tring/PycharmProjects/Capstone-Stuff/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

  X Motor Encoder Connector ID (on Connector C)
- #define **Y_CONNECTOR_ID** 1

  Y Motor Encoder Connector ID (on Connector C)
- #define POTENTIOMETER_SLOPE -2.11 ∗ PI / 180.0
- #define **POT_V_LIM_LO** -20.0

  Lower Potentiometer Voltage Saturation Limit (V)
- #define **POT_V_LIM_HI** 20.0

  Upper Potentiometer Voltage Saturation Limit (V)
- #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

  Lower X Limit.
- #define **Y_LIM_LO** 0.0

  Lower Y Limit.
- #define **X_LIM_HI** 0.35

  Higher X Limit.
- #define **Y_LIM_HI** 0.35

Higher Y Limit.

- #define **VEL_LIM_ABS** 1.0

  Absolute Velocity Limit.
- #define **CHANNELS** 16

  Number of Channels.
- #define **LCD_KEYPAD_LEN** 4

  Keypad Length.
- #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**

  Reset Variable for measuring velocity.
- static float potentiometer_v_x_intercept
- static float potentiometer_v_y_intercept
- static MyRio_Aio **x_potentiometer**

  X Potentiometer.
- static MyRio_Aio **y_potentiometer**

  Y Potentiometer.
- static MyRio_Encoder **x_encoder**

  X Motor Encoder.

- static MyRio_Encoder **y_encoder**
  Y Motor Encoder.
- static int32_t first_enc_state [2]
- static int32_t prev_enc_state [2]
- static bool **holding_vel_set**
  Indicator if the holding for velocity is set.
- static bool **holding_pos_set**
  Indicator if the holding for position is set.
- static Velocities **holding_vel**
  Encoder Holding for velocity.
- static Positions **holding_pos**
  Encoder Holding for position.
- static const Encoder_StatusMask enc_st_mask
  Encoder Error Mask.
- MyRio_Aio **x_motor**
  X Motor Voltage Channel.
- MyRio_Aio **y_motor**
  Y Motor Voltage Channel.
- MyRio_IrqTimer timer
  Universal Timer.
- static MyRio_Dio **channel** [CHANNELS]
  Keyboard channels.
- static pthread_mutex_t **keyboard**
  Keyboard lock.
- static Keymap **keymap**
  Our keymap.
- static pthread_t **keymap_thread**
  Thread for Keymap Thread.
- static ThreadResource **keymap_resource**
  Thread Resource for Keymap Thread.
- static int **error**
  Local Error Flag.

4.13.1 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

4.13.2 Macro Definition Documentation

4.13.2.1 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

4.13.2.2 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

value	The value to convert

4.13.2.3 ENC_CNT_REV

```
#define ENC_CNT_REV 2000.0
```

Number of counts in one revolution TODO(nguy8tri): Find this quantity

4.13.2.4 M_PER_REV

```
#define M_PER_REV 0.01267 * PI
```

Meters per revolution Diameter of upper pulley (12 mm) $*$ PI

4.13.2.5 POTENTIOMETER_SLOPE

```
#define POTENTIOMETER_SLOPE -2.11 * PI / 180.0
```

Best-Fit Potentiometer Slope (rad/V) TODO(nguy8tri): Find this quantity

4.13.2.6 UNIT_VEL

```
#define UNIT_VEL 0.15
```

The unit velocity stop corresponding to a keypad touch (m/s)

4.13.3 Typedef Documentation

4.13.3.1 Keymap

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

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

4.13.4 Function Documentation

4.13.4.1 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:

4.13.4.2 getkey()

```
char getkey ()
```

Keypad characters

Locking style allows for getkey() to take precedence over all other keyboard commandsHere is the call graph for this function:

4.13.4.3 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

4.13.4.4 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

Setup discrete velocity commands, -1, 0, and 1

4.13.4.5 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:

4.13.4.6 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:

4.13.4.7 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

4.13.4.8 GetUserVelocity()

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

Obtains the User Velocity

Parameters

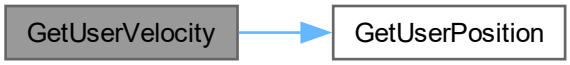
angle	The rope angle
vel	The trolley velocity
result	A return parameter, which will become the velocity of the user

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:



4.13.4.9 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

Check Positional Limits first

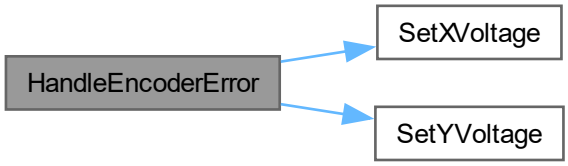
Now, check velocity limits

Now, check if there is an encoder error

u_error = EENCR; conC_Encoder_initialize(myrio_session, &x_encoder, X_CONNECTOR_ID);

u_error = EENCR; conC_Encoder_initialize(myrio_session, &y_encoder, Y_CONNECTOR_ID);

Output ErrorHere is the call graph for this function:



4.13.4.10 HandlePotentiometerError()

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

Handles Error Processing for Potentiometer Measurements

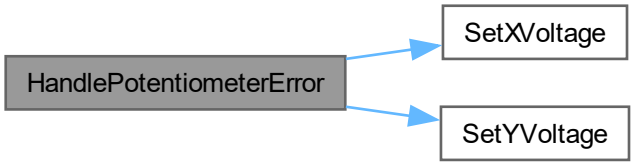
Parameters

curr_ang	The current angle reading

Returns

0 upon no error, ESTRN otherwise

Here is the call graph for this function:



4.13.4.11 IOSetup()

```
int IOSetup ()
```

Sets up the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

Setup Timer

Setup Encoders Channels

Setup Potentiometer Voltage Channels (are swapped)

Setup Motor Channels

Setup Keyboard Channels & Resources

Setup Reset flag

Calibration Message

Set Reference Positions

Setup the holding

Calibrate voltage intercepts for potentiometerHere is the call graph for this function:

4.13.4.12 IOShutdown()

```
int IOShutdown ()
```

Shutsdown the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

Dissasociate with Encoders

Dissasociate with Potentiometers

Disassociate with Motor

Destroy Keyboard Lock

4.13.4.13 KeyboardControlFork()

```
int KeyboardControlFork ()
```

Enables Keyboard Control for Anti-Sway (concurrently)

Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Begin Keyboard ThreadHere is the call graph for this function:

4.13.4.14 KeyboardControlJoin()

```
int KeyboardControlJoin ()
```

Stops Keyboard Control for Anti-Sway (concurrent process)

Returns

0 upon success, negative if error

Destroy Keymap Thread

4.13.4.15 KeymapThread()

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

Obtains the numerical buttons pressed (1 through 9)

Returns

NULL

Postcondition

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

4.13.4.16 PressedDelete()

```
bool PressedDelete ()
```

Detects if the DEL key is pressed on the keyboard

Returns

true iff DEL is pressed on the keyboard

4.13.4.17 Reset()

```
void Reset ()
```

Resets GetTrolleyPosition and GetTrolleyVelocity by setting the velocity to zero

Postcondition

The next time GetTrolleyVelocity is called, both velocities are zero

4.13.4.18 SetXVoltage()

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

Sets the voltage of the X motor

Returns

0 upon success, other integers if otherwise

4.13.4.19 SetYVoltage()

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

Sets the voltage of the Y motor

Returns

0 upon success, other integers if otherwise

4.13.4.20 wait()

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

Waits for approximate 5 ms

Postcondition

About 5 ms have passed

Wait Constant

4.13.5 Variable Documentation

4.13.5.1 enc_st_mask

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

Initial value:

```
=
    (Encoder_StError)
```

Encoder Error Mask.

4.13.5.2 first_enc_state

```
int32_t first_enc_state[2]  [static]
```

First Encoder state for both the X and Y Encoders

4.13.5.3 potentiometer_v_x_intercept

```
float potentiometer_v_x_intercept  [static]
```

Calibrated Voltage Intercept (x-intercept) for X Potentiometer

4.13.5.4 potentiometer_v_y_intercept

```
float potentiometer_v_y_intercept  [static]
```

Calibrated Voltage Intercept (x-intercept) for Y Potentiometer

4.13.5.5 prev_enc_state

```
int32_t prev_enc_state[2]  [static]
```

Previous Encoder state (from the last time either GetTrolleyPosition or GetTrolleyVelocity is caled), for both the X and Y Encoders

4.13.5.6 timer

```
MyRio_IrqTimer timer
```

Universal Timer.

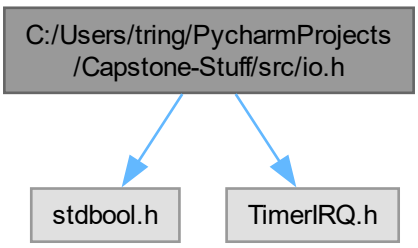
The Timer.

4.14 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/io.h File Reference

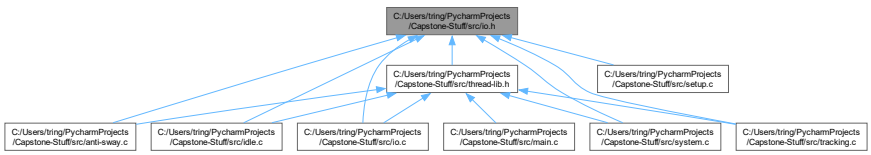
Sensor/Actuator (Input/Output) Interfacing Library Header.

```
#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
  Motor Voltage High Limit (V)
- #define **MOTOR_V_LIM_L** -10.000
  Motor Voltage Low Limit (V)
- #define **R** 0.0062
  Pulley Radius (m)
- #define **K_a** 0.41
  Current Constant (A/V)
- #define **K_m** 0.11
  Motor Constant (Nm/A)
- #define FORCE_TO_VOLTAGE(force) (force) ∗ R / (K_a ∗ K_m)
  Force to Voltage Conversion.
- #define VOLTAGE_TO_FORCE(voltage) (voltage) ∗ (K_a ∗ K_m) / R
  Voltage to Force Conversion.

Typedefs

- typedef float **Angle**
  Alias for an Angle.
- typedef float **Position**
  Alias for a Position.
- typedef float **Velocity**
  Alias for Velocity.
- typedef float **Voltage**
  Alias for 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
  The Timer.

4.14.1 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

4.14.2 Macro Definition Documentation

4.14.2.1 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 (through the motor)

Postcondition

Becomes the conversion between force to the voltage to output

4.14.2.2 VOLTAGE_TO_FORCE

```
#define VOLTAGE_TO_FORCE(
              voltage)  (voltage) * (K_a * K_m) / R
```

Voltage to Force Conversion.

Parameters

voltage	An int/float/double expression, which represents the voltage to transmit (through the motor)

Postcondition

Converts voltage into a force

4.14.3 Function Documentation

4.14.3.1 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:

4.14.3.2 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

4.14.3.3 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

Setup discrete velocity commands, -1, 0, and 1

4.14.3.4 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:

4.14.3.5 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:

4.14.3.6 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

4.14.3.7 GetUserVelocity()

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

Obtains the User Velocity

Parameters

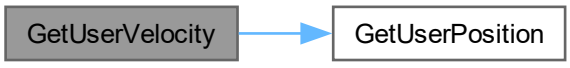
angle	The rope angle
vel	The trolley velocity
result	A return parameter, which will become the velocity of the user

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:



4.14.3.8 IOSetup()

```
int IOSetup ()
```

Sets up the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

Setup Timer

Setup Encoders Channels

Setup Potentiometer Voltage Channels (are swapped)

Setup Motor Channels

Setup Keyboard Channels & Resources

Setup Reset flag

Calibration Message

Set Reference Positions

Setup the holding

Calibrate voltage intercepts for potentiometerHere is the call graph for this function:

4.14.3.9 IOShutdown()

```
int IOShutdown ()
```

Shutsdown the System-Sensor/Actuator Interface

Returns

0 upon success, negative otherwise

Dissasociate with Encoders

Dissasociate with Potentiometers

Disassociate with Motor

Destroy Keyboard Lock

4.14.3.10 KeyboardControlFork()

```
int KeyboardControlFork ()
```

Enables Keyboard Control for Anti-Sway (concurrently)

Postcondition

If its already running, does nothing

Returns

0 upon success, negative if error

Begin Keyboard ThreadHere is the call graph for this function:

4.14.3.11 KeyboardControlJoin()

```
int KeyboardControlJoin ()
```

Stops Keyboard Control for Anti-Sway (concurrent process)

Returns

0 upon success, negative if error

Destroy Keymap Thread

4.14.3.12 PressedDelete()

```
bool PressedDelete ()
```

Detects if the DEL key is pressed on the keyboard

Returns

true iff DEL is pressed on the keyboard

4.14.3.13 Reset()

```
void Reset ()
```

Resets GetTrolleyPosition and GetTrolleyVelocity by setting the velocity to zero

Postcondition

The next time GetTrolleyVelocity is called, both velocities are zero

4.14.3.14 SetXVoltage()

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

Sets the voltage of the X motor

Returns

0 upon success, other integers if otherwise

4.14.3.15 SetYVoltage()

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

Sets the voltage of the Y motor

Returns

0 upon success, other integers if otherwise

4.14.4 Variable Documentation

4.14.4.1 timer

```
MyRio_IrqTimer timer  [extern]
```

The Timer.

The Timer.

4.15 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
00024 typedef float Angle;
00026 typedef float Position;
00028 typedef float Velocity;
00030 typedef float Voltage;
00031
00032
00039 typedef struct {
00041     Angle x_angle;
00043     Angle y_angle;
00044 } Angles;
00045
00052 typedef struct {
00054     Position x_pos;
00056     Position y_pos;
00057 } Positions;
00058
00065 typedef struct {
00067     Velocity x_vel;
00069     Velocity y_vel;
00070 } Velocities;
00071
00072 /* Sensor Variables */
00074 extern MyRio_IrqTimer timer;
00075
00076 /* Actuator Limits */
00078 #define MOTOR_V_LIM_H 10.000
00080 #define MOTOR_V_LIM_L -10.000
00081
00082 /* Physical Parameters */
00084 #define R 0.0062
00086 #define K_a 0.41
00088 #define K_m 0.11
00099 #define FORCE_TO_VOLTAGE(force) \
00100     (force) * R / (K_a * K_m)
00110 #define VOLTAGE_TO_FORCE(voltage) \
00111     (voltage) * (K_a * K_m) / R
00112
00113 /* Setup/Shutdown Functions */
00114
00115
00123 int IOSetup();
00124
00132 int IOShutdown();
00133
00134
00135 /* Reset Feature */
00136
00137
00146 void Reset();
00147
00148
00149 /* Sensor Functions */
00150
00151
00163 int GetReferenceVelocityCommand(Velocities *result);
00164
00176 int GetReferenceAngleCommand(Angles *result);
00177
00190 int GetAngle(Angles *result);
00191
00206 int GetTrolleyPosition(Positions *result);
00207
00222 int GetTrolleyVelocity(Velocities *result);
00223
00238 int GetUserPosition(Angles *angle, Positions *pos, Positions *result);
00239
00254 int GetUserVelocity(Angles *angle, Velocities *vel, Velocities *result);
00255
00256
00257 /* Actuator Functions */
00258
```

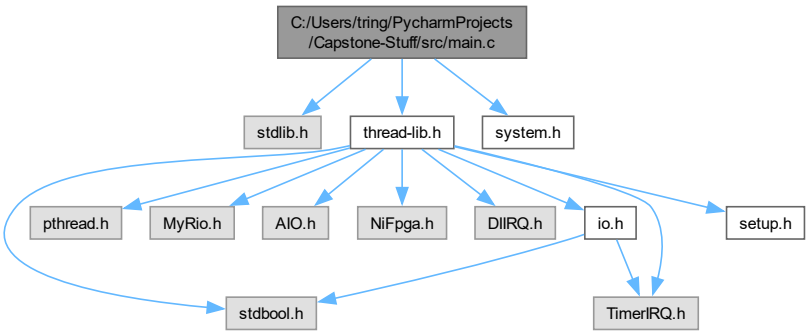
```
00259
00266 int SetXVoltage(Voltage voltage);
00267
00274 int SetYVoltage(Voltage voltage);
00275
00276
00277 /* Keyboard Functions */
00278
00286 bool PressedDelete();
00287
00295 int KeyboardControlFork();
00296
00302 int KeyboardControlJoin();
00303
00304 #endif
```

4.16 C:/Users/tring/PycharmProjects/Capstone-Stuff/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)

4.16.1 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

4.16.2 Function Documentation

4.16.2.1 main()

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

Runs the Anti-Sway Capstone Project

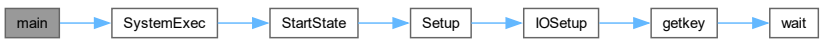
Parameters

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

Returns

0 iff success, negative otherwise

Here is the call graph for this function:



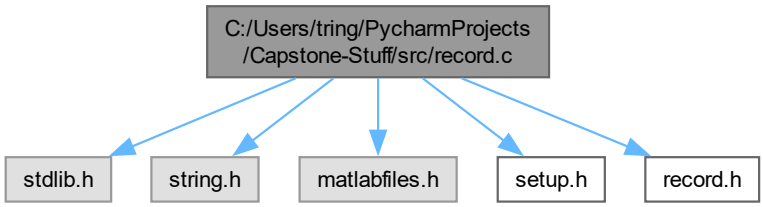
4.17 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/record.c File Reference

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

  Default number of Files to remember.
- #define **DEFAULT_NUM_VALS** 10

  Default number of values for data arrays.
- #define **DEFAULT_RESIZE_FACTOR** 2

  Default resize factor.

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

  The data files this module is handling.
- static int **num_files** = 0

  The number of files this module is handling.
- static int **capacity_files** = 0

  The number of files this module can handle.

4.17.1 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

4.17.2 Function Documentation

4.17.2.1 DeallocateHelper()

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

Deallocates the entire module

4.17.2.2 OpenDataFile()

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

Opens a data file

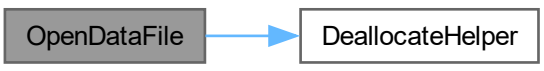
Parameters

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:



4.17.2.3 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

4.17.2.4 RecordData()

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

Records data for each entry

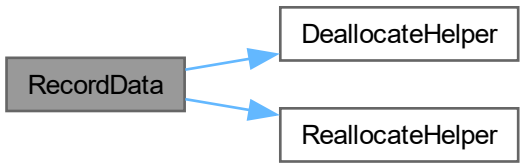
Parameters

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:



4.17.2.5 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

4.17.2.6 SaveDataFiles()

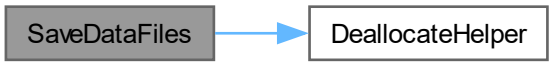
```
int SaveDataFiles ()
```

Records all data into actual files, and closes all files

Returns

0 iff success, negative upon failure

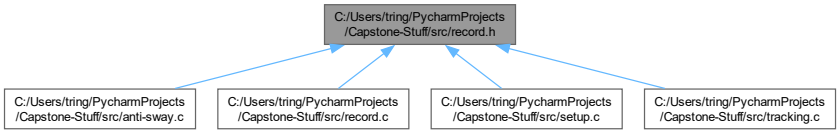
Here is the call graph for this function:



4.18 C:/Users/tring/PycharmProjects/Capstone-Stuff/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**
  A File.

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 ()

4.18.1 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

4.18.2 Function Documentation

4.18.2.1 OpenDataFile()

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

Opens a data file

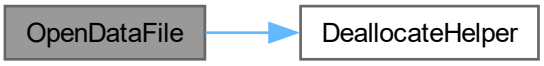
Parameters

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:



4.18.2.2 RecordData()

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

Records data for each entry

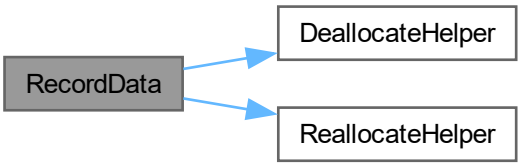
Parameters

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:



4.18.2.3 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

4.18.2.4 SaveDataFiles()

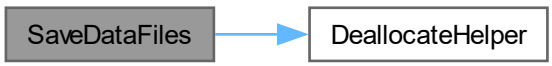
```
int SaveDataFiles ()
```

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:



4.19 record.h

Go to the documentation of this file.

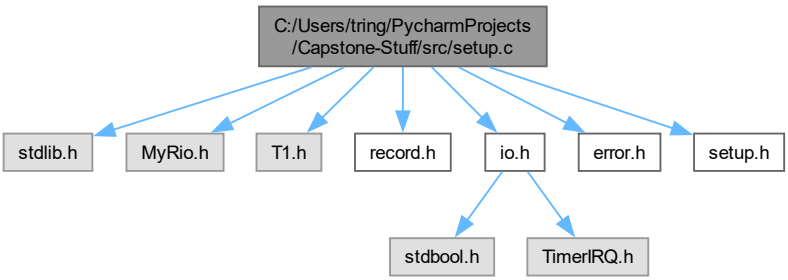
```
00001
00013 #ifndef RECORD_H_
00014 #define RECORD_H_
00015
00017 typedef int FileID_t;
00018
00028 FileID_t OpenDataFile(char *name, char **entry_names, int num_entries);
00029
00039 int RecordData(FileID_t file, double data[], int data_length);
00040
00050 int RecordValue(FileID_t file, char *value_name,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

4.20 C:/Users/tring/PycharmProjects/Capstone-Stuff/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**

  Local error flag.
- int u_error

  Universal error code (extern)

4.20.1 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

4.20.2 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()

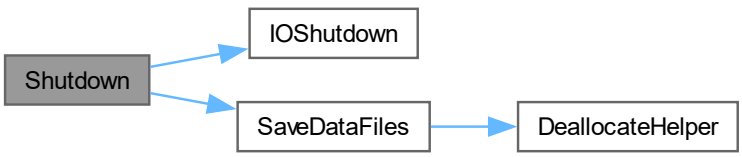
```
int Shutdown ()
```

Shuts the entire System down

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.20.3 Variable Documentation

4.20.3.1 u_error

```
int u_error
```

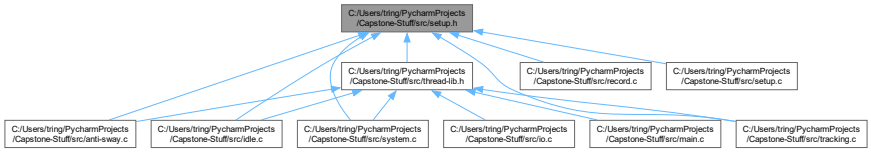
Universal error code (extern)

Ther universal error code.

4.21 C:/Users/tring/PycharmProjects/Capstone-Stuff/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
  Verifies that a statement is true.

Functions

- int Setup ()
- int Shutdown ()

4.21.1 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

4.21.2 Macro Definition Documentation

4.21.2.1 VERIFY

```
#define VERIFY(
               code,
               statement)  if ((code = statement)) return code
```

Verifies that a statement is true.

Parameters

code	An integer varibale to hold the value of statement
statement	The statement to verify

Returns

false if statement is false

Postcondition

If a return is not executed, then code is now 0

4.21.3 Function Documentation

4.21.3.1 Setup()

```
int Setup ()
```

Sets up the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:

4.21.3.2 Shutdown()

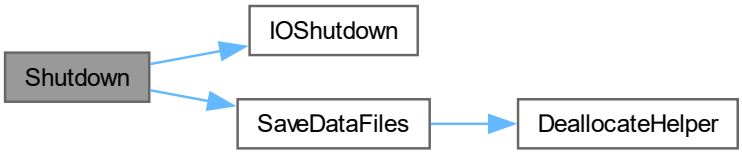
```
int Shutdown ()
```

Shuts the entire System down

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.22 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
00031 #define VERIFY(code, statement) \
00032     if ((code = statement)) return code
00033
00034
00035 /* Global Setup/Shutdown Functions */
00036
00037
00044 int Setup();
00045
00046
00053 int Shutdown();
00054
00055 #endif  // SETUP_H_
```

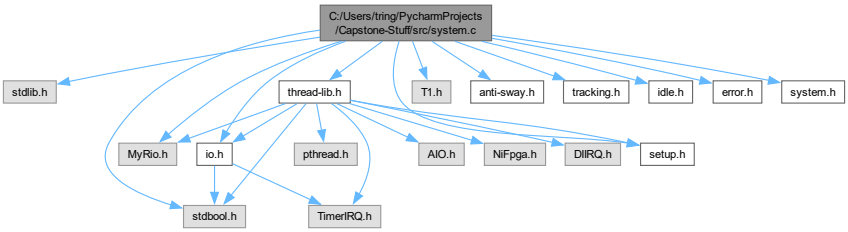
4.23 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/system.c File Reference

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 [ ])()

  State Functions.
- static States **state** = START

  Current State.
- static int **error**

  Local Error Code.

4.23.1 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

4.23.2 Enumeration Type Documentation

4.23.2.1 States

```
enum States
```

The possible states

4.23.3 Function Documentation

4.23.3.1 AntiSwayState()

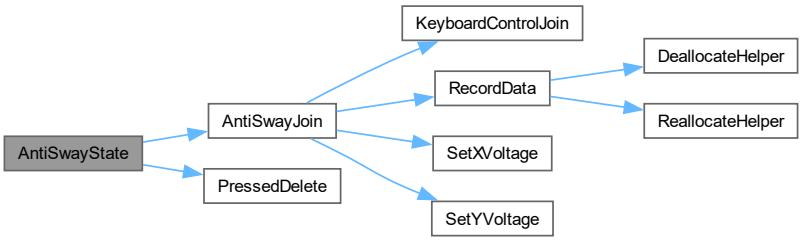
```
static int AntiSwayState ()  [static]
```

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:



4.23.3.2 EndState()

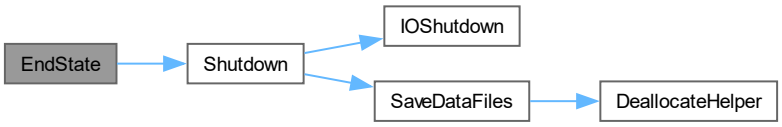
```
static int EndState ()  [static]
```

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:



4.23.3.3 ErrorState()

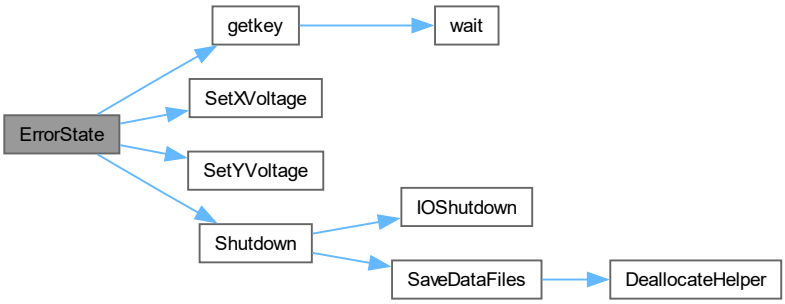
```
static int ErrorState ()  [static]
```

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:



4.23.3.4 IdleState()

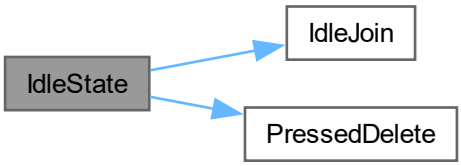
```
static int IdleState ()  [static]
```

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:



4.23.3.5 MenuState()

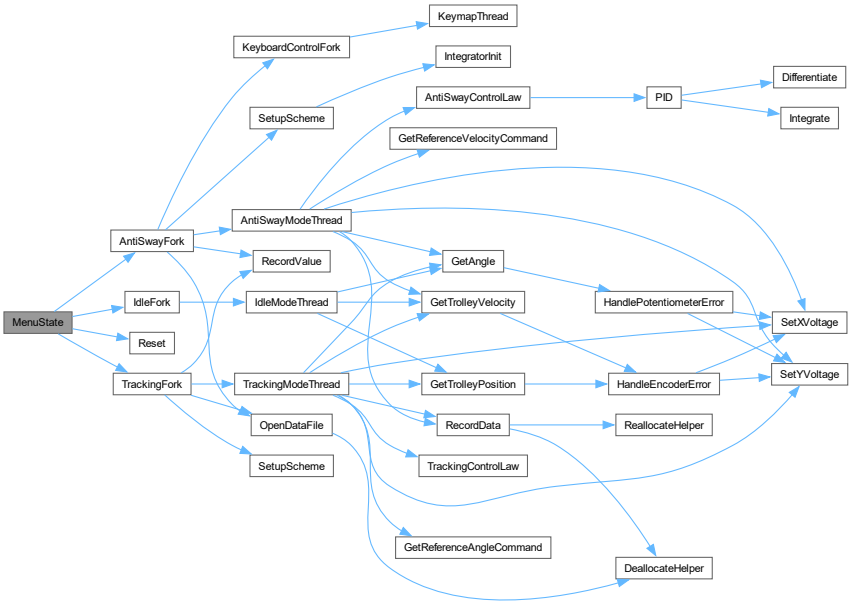
```
static int MenuState ()  [static]
```

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:



4.23.3.6 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:

4.23.3.7 SystemExec()

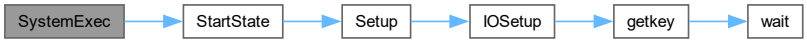
```
int SystemExec ()
```

Executes the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:



4.23.3.8 TrackingState()

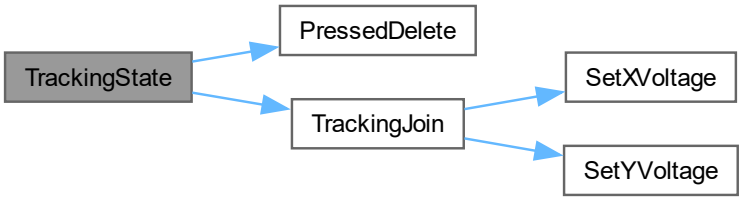
```
static int TrackingState ()  [static]
```

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:



4.23.4 Variable Documentation

4.23.4.1 states

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

Initial value:

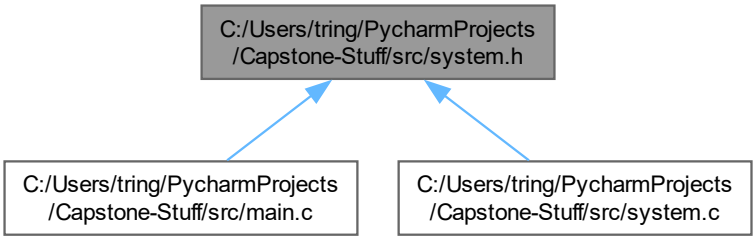
```
= {AntiSwayState,
                                TrackingState,
                                IdleState,
                                MenuState,
                                ErrorState,
                                StartState,
                                EndState}
```

State Functions.

4.24 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/system.h File Reference

System (Turing Machine) Header.

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



Functions

- int SystemExec ()

4.24.1 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

Copyright (c) 2024

4.24.2 Function Documentation

4.24.2.1 SystemExec()

```
int SystemExec ()
```

Executes the entire System

Returns

0 upon success, negative otherwise

Here is the call graph for this function:

4.25 system.h

Go to the documentation of this file.

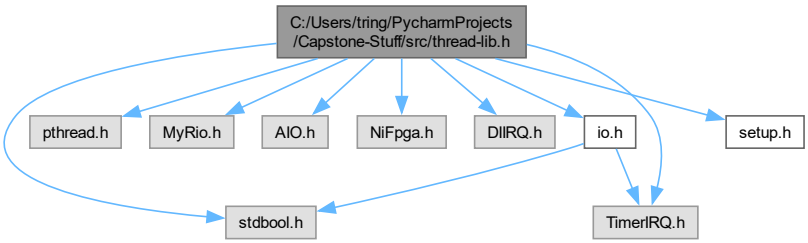
```
00001
00013 #ifndef SYSTEM_H_
00014 #define SYSTEM_H_
00015
00016
00017 /* Execution Function */
00018
00019
00026 int SystemExec();
00027
00028 #endif  // SYSTEM_H_
```

4.26 C:/Users/tring/PycharmProjects/Capstone-Stuff/src/thread-lib.h File Reference

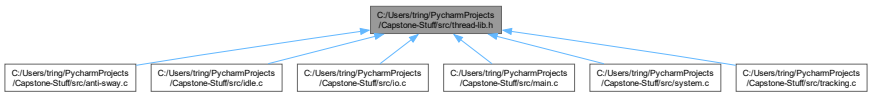
Thread Library.

```
#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
  The timestep, in microseconds (us)
- #define **BTI_MS** 5u
  The timestep, in milliseconds (ms)
- #define **BTI_S** 0.005
  The timestep, in seconds (s)
- #define **g** 9.81
  Acceleration due to Gravity (m/s^ 2)
- #define **PI** 3.141592653549
  Pi.
- #define **l** 0.47
  Length of Rope (m)
- #define **m_dt** 2.092
  Mass of the double Trolley (kg)
- #define **m_st** 0.664
  Mass of the single Trolley (kg)
- #define **m_p** 0.765
  Mass of User (kg)
- #define START_THREAD(thread, function, resource)
  Starts a thread.
- #define REGISTER_TIMER(resource) Irq_RegisterTimerIrq(&timer, &(resource.irq_context), BTI_US)
  Registers the global timer with a thread.
- #define STOP_THREAD(thread, resource)
  Stops a thread in this process.
- #define UNREGISTER_TIMER(resource) Irq_UnregisterTimerIrq(&timer, resource.irq_context)
  Unregisters the global timer with a thread.
- #define TIMER_TRIGGER(irq_assert, resource)
- #define EXIT_THREAD()

Variables

- NiFpga_Session **myrio_session**
  The MyRio Session.

4.26.1 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

4.26.2 Macro Definition Documentation

4.26.2.1 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.

4.26.2.2 REGISTER_TIMER

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

Registers the global timer with a thread.

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

Parameters

resource	The ThreadResource associated with a thread

Precondition

No other thread is using the global timer

Postcondition

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

4.26.2.3 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.

Starts a new thread within this process.

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

4.26.2.4 STOP_THREAD

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

Value:

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

Stops a thread in this process.

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

4.26.2.5 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)

4.26.2.6 UNREGISTER_TIMER

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

Unregisters the global timer with a thread.

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

4.27 thread-lib.h

Go to the documentation of this file.

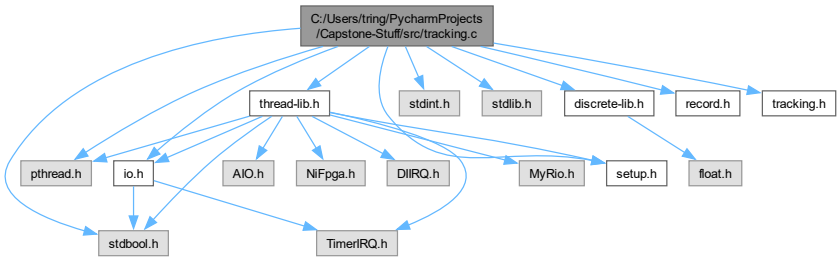
```
00001
00012 #ifndef THREAD_LIB_H_
00013 #define THREAD_LIB_H_
00014
00015 #include <stdbool.h>
00016 #include <pthread.h>
00017
00018 #include "MyRio.h"
00019 #include "AIO.h"
00020 #include "NiFpga.h"
00021 #include "DIIRQ.h"
00022 #include "TimerIRQ.h"
00023 #include "io.h"
00024
00025 #include "setup.h"
00026
00027
00028 /* Thread Data Structures */
00029
00030
00036 typedef struct {
00037     NiFpga_IrqContext irq_context;
00038     NiFpga_Bool irq_thread_rdy;
00039 } ThreadResource;
00040
00041
00042 /* Time Constants */
00043
00044
00046 #define BTI_US 5000u
00048 #define BTI_MS 5u
00050 #define BTI_S 0.005
00051
00052
00053 /* Physical Constants */
00054
00055
00057 #define g 9.81
00059 #define PI 3.141592653549
00061 #define l 0.47
00063 #define m_dt 2.092
00065 #define m_st 0.664
00067 #define m_p 0.765
00068
00069
00070 /* MyRio Session */
00071
00073 extern NiFpga_Session myrio_session;
00074
00075
00076 /* Thread Construction/Destruction */
00077
00078
00094 #define START_THREAD(thread, function, resource) \
00095     resource.irq_thread_rdy = true; \
00096     VERIFY(error, pthread_create(&thread, NULL, function, &resource))
00097
00108 #define REGISTER_TIMER(resource) \
00109     Irq_RegisterTimerIrq(&timer, &(resource.irq_context), BTI_US)
00110
00126 #define STOP_THREAD(thread, resource) \
00127     resource.irq_thread_rdy = false; \
00128     VERIFY(error, pthread_join(thread, NULL))
00129
00139 #define UNREGISTER_TIMER(resource) \
00140     Irq_UnregisterTimerIrq(&timer, resource.irq_context)
00141
00153 #define TIMER_TRIGGER(irq_assert, resource) \
00154     Irq_Wait(resource->irq_context, \
00155              TIMERIRQNO, \
00156              &irq_assert, \
00157              (NiFpga_Bool *) &(resource->irq_thread_rdy)); \
00158     NiFpga_WriteU32(myrio_session, IRQTIMERWRITE, BTI_US); \
00159     NiFpga_WriteBool(myrio_session, IRQTIMERSETTIME, NiFpga_True)
00160
00167 #define EXIT_THREAD() \
00168     pthread_exit(NULL); \
00169     return NULL
00170
00171 #endif  // THREAD_LIB_H_
```

4.28 C:/Users/tring/PycharmProjects/Capstone-Stuff/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

  Reference Angle (rad)
- #define **T_s** 0.1

  The settling time (0.1 s)
- #define **os** 0.05

  The overshoot fraction (5%)
- #define **DATA_LEN** 12

  The number of entries.

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**
  Thread ID.
- ThreadResource **resource**
  Thread Resources (Shared Resources)
- static TrackingControlScheme **x_control**
  The Control Scheme for the X Motor.
- static TrackingControlScheme **y_control**
  The Control Scheme for the Y Motor.
- static int **error**
  Local Error Flag.
- static FileID_t **file** = -1
  The file ID.
- static char * **data_file_name** = "tracking.mat"
  The file Name.
- static char * data_names [DATA_LEN]
  The data names.
- static double **data** [DATA_LEN]
  Buffer for data.
- static double * **data_buff** = data
  Pointer to next data point to insert into buffer.
- static int **id** = 1
  ID variable.

4.28.1 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

Copyright (c) 2024

4.28.2 Function Documentation

4.28.2.1 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

4.28.2.2 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

4.28.2.3 TrackingFork()

```
int TrackingFork ()
```

Executes Tracking Mode (concurrently)

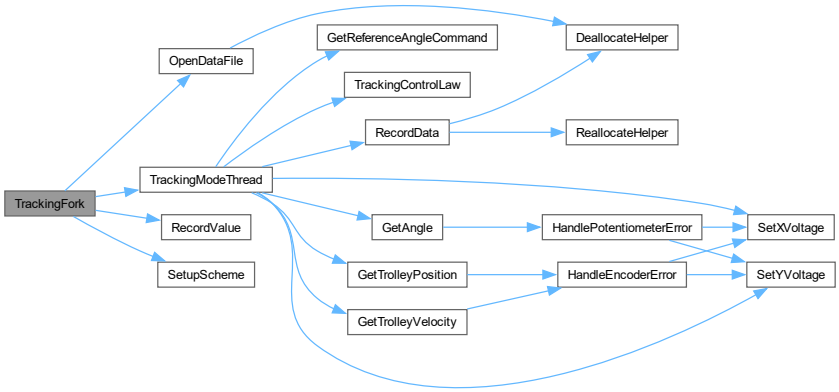
Precondition

Tracking Mode is not already running

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.28.2.4 TrackingJoin()

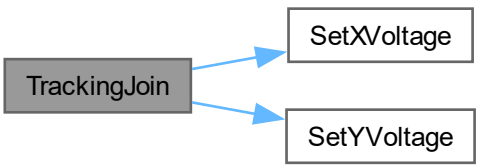
```
int TrackingJoin ()
```

Stops Tracking Mode (concurrent process)

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.28.2.5 TrackingModeThread()

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

The Thread Function for Tracking Mode

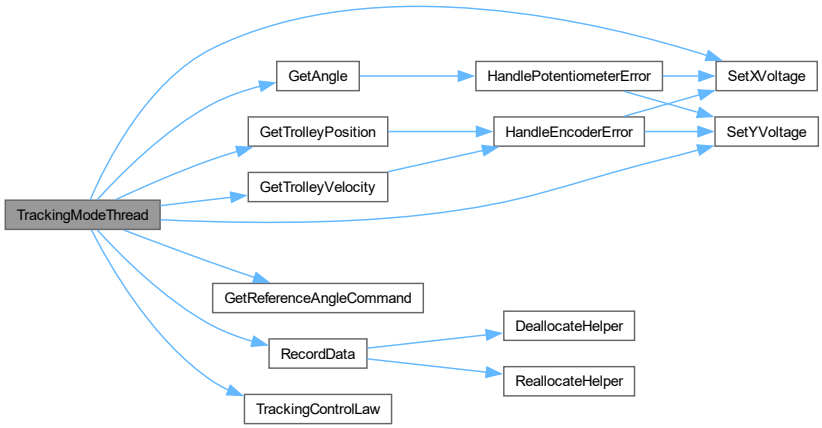
Parameters

resource	A pointer to a Resource sturcture for Tracking Mode

Returns

NULL

Here is the call graph for this function:



4.28.3 Variable Documentation

4.28.3.1 data_names

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

Initial value:

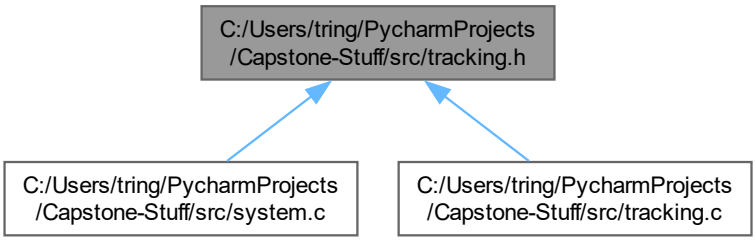
```
= {"id", "t",
                                    "angle_x", "angle_y",
                                    "trolley_pos_x", "trolley_pos_y",
                                    "trolley_vel_x", "trolley_vel_y",
                                    "inner_x", "voltage_x",
                        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

The data names.

4.29 C:/Users/tring/PycharmProjects/Capstone-Stuff/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 ()

4.29.1 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

4.29.2 Function Documentation

4.29.2.1 TrackingFork()

```
int TrackingFork ()
```

Executes Tracking Mode (concurrently)

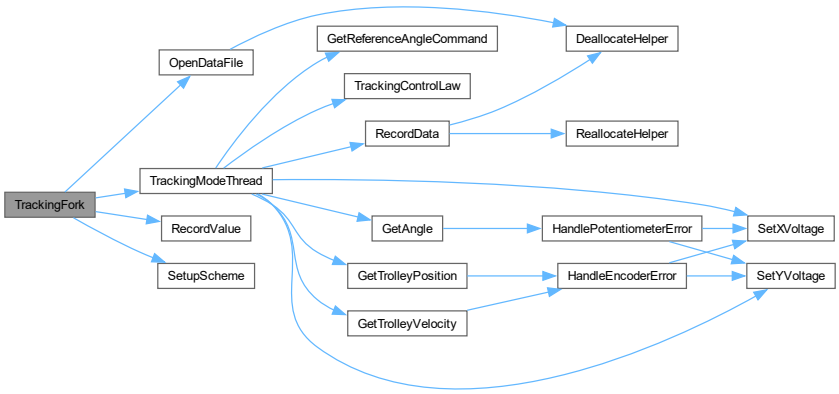
Precondition

Tracking Mode is not already running

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.29.2.2 TrackingJoin()

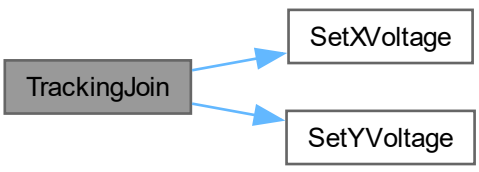
```
int TrackingJoin ()
```

Stops Tracking Mode (concurrent process)

Returns

0 upon success, negative if error

Here is the call graph for this function:



4.30 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_
```