IDPL108

1.0

Generated by Doxygen 1.7.1

Thu Feb 3 2011 17:50:22

Contents

1 Main Page **1**

2 Namespace Index **3**

2.1 Namespace List 3

3 Class Index **5**

3.1 Class List 5

4 File Index **7**

4.1 File List 7

5 Namespace Documentation **9**

5.1 IDP Namespace Reference 9

5.1.1 Enumeration Type Documentation 11

5.1.1.1 BobbinBadness 11

5.1.1.2 BobbinColour 11

5.1.1.3 LineFollowingStatus 11

5.1.1.4 LineSensorStatus 11

5.1.1.5 NavigationLocation 12

5.1.1.6 NavigationStatus 12

5.1.2 Function Documentation 12

5.1.2.1 cap_correction 12

5.1.3 Variable Documentation 12

5.1.3.1 EDGE_ERROR 12

5.1.3.2 INTEGRAL_GAIN 12

5.1.3.3 LOST_TIMEOUT 12

5.1.3.4 MAX_CORRECTION 13

5.1.3.5 MOTOR_MAX_SPEED 13

5.1.3.6 MOTOR_RAMP_TIME 13

6 Class Documentation 15

6.1 IDP::ClampControl Class Reference 15

6.1.1 Detailed Description 16

6.1.2 Constructor & Destructor Documentation 16

6.1.2.1 ClampControl 16

6.1.3 Member Function Documentation 16

6.1.3.1 badness 16

6.1.3.2 colour 16

6.1.3.3 pick_up 16

6.1.3.4 put_down 17

6.2 IDP::HardwareAbstractionLayer Class Reference 17

6.2.1 Detailed Description 18

6.2.2 Constructor & Destructor Documentation 18

6.2.2.1 HardwareAbstractionLayer 18

6.2.3 Member Function Documentation 19

6.2.3.1 bad_bobbin_ldr 19

6.2.3.2 bad_bobbin_led 19

6.2.3.3 clear_status_register 19

6.2.3.4 colour_ldr 19

6.2.3.5 colour_leds 19

6.2.3.6 grabber_jaw 19

6.2.3.7 grabber_lift 20

6.2.3.8 grabber_switch 20

6.2.3.9 indication_LEDs 20

6.2.3.10 line_following_sensors 20

6.2.3.11 motor_left_backward 20

6.2.3.12 motor_left_forward 20

6.2.3.13 motor_right_backward 21

6.2.3.14 motor_right_forward 21

6.2.3.15 motors_backward 21

6.2.3.16 motors_forward 21

6.2.3.17 motors_stop 21

6.2.3.18 motors_turn_left 21

6.2.3.19 motors_turn_right 22

6.2.3.20 reset_switch 22

6.2.3.21 status_register 22

6.3 IDP::LineFollowing Class Reference 22
6.3.1 Detailed Description 23
6.3.2 Constructor & Destructor Documentation 23
6.3.2.1 LineFollowing 23
6.3.3 Member Function Documentation 24
6.3.3.1 follow_line 24
6.3.3.2 set_speed 24
6.3.3.3 turn_left 24
6.3.3.4 turn_right 24
6.4 IDP::LineSensors Struct Reference 24
6.4.1 Detailed Description 24
6.4.2 Member Data Documentation 25
6.4.2.1 line_left 25
6.4.2.2 line_right 25
6.4.2.3 outer_left 25
6.4.2.4 outer_right 25
6.5 IDP::MissionSupervisor Class Reference 25
6.5.1 Detailed Description 26
6.5.2 Constructor & Destructor Documentation 26
6.5.2.1 MissionSupervisor 26
6.5.3 Member Function Documentation 26
6.5.3.1 drive_backward 26
6.5.3.2 drive_forward 26
6.5.3.3 hal 26
6.5.3.4 run_task 26
6.5.3.5 stop 27
6.5.3.6 test_line_following 27
6.5.3.7 test_line_sensor 27
6.6 IDP::Navigation Class Reference 27
6.6.1 Detailed Description 28
6.6.2 Constructor & Destructor Documentation 28
6.6.2.1 Navigation 28
6.6.3 Member Function Documentation 28
6.6.3.1 go 28
6.7 IDP::SelfTests Class Reference 28
6.7.1 Detailed Description 30

6.7.2 Constructor & Destructor Documentation 30
6.7.2.1 SelfTests 30
6.7.3 Member Function Documentation 30
6.7.3.1 actuators 30
6.7.3.2 badness_LED 30
6.7.3.3 bobbin_analyse 30
6.7.3.4 clamp_control 30
6.7.3.5 colour_sensor_LEDs 30
6.7.3.6 drive_backward 30
6.7.3.7 drive_forward 30
6.7.3.8 LDRs 30
6.7.3.9 line_following 31
6.7.3.10 line_sensors 31
6.7.3.11 microswitches 31
6.7.3.12 navigate 31
6.7.3.13 position 31
6.7.3.14 status_LEDs 31
6.7.3.15 steer_left 31
6.7.3.16 steer_right 31
6.7.3.17 stop 31
6.7.3.18 turn_left 31
6.7.3.19 turn_right 31
6.8 IDP::StatusWatchdog Class Reference 32
6.8.1 Detailed Description 32
6.8.2 Member Function Documentation 32
6.8.2.1 check 32

7 File Documentation 33
7.1 libidp/clamp_control.cc File Reference 33
7.2 libidp/clamp_control.h File Reference 34
7.3 libidp/hal.cc File Reference 34
7.4 libidp/hal.h File Reference 35
7.5 libidp/libidp.h File Reference 36
7.6 libidp/line_following.cc File Reference 37
7.7 libidp/line_following.h File Reference 38
7.8 libidp/mission_supervisor.cc File Reference 39
7.9 libidp/mission_supervisor.h File Reference 40

7.10 libidp/navigation.cc File Reference 41
7.11 libidp/navigation.h File Reference 41
7.12 libidp/self_tests.cc File Reference 42
7.13 libidp/self_tests.h File Reference 43
7.14 libidp/status_watchdog.cc File Reference 44
7.15 libidp/status_watchdog.h File Reference 44

Chapter 1

Main Page

Integrated Design Project Team L108 Source Code Documentation. See IDP Namespace for API.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

IDP 9

Chapter 3

Class Index

3.1 Class List

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

IDP::ClampControl (Manage the actuation of the clamp, as well as the detection and analysis of bobbins for their colour and badness) 15
IDP::HardwareAbstractionLayer (Provide a hardware agnostic interface to the required hardware functionality) 17
IDP::LineFollowing (Maintain the robot position correctly with respect to the white line markers, during driving and manouvering) 22
IDP::LineSensors (Contains the LINE or NO_LINE status of each of the four IR sensors used for line following) 24
IDP::MissionSupervisor (Control the overall robot behaviour and objective fulfillment) 25
IDP::Navigation (Find a route from one place to another on the board, and maintain an estimate of the current position) 27
IDP::SelfTests (Execute a variety of functionality self tests) 28
IDP::StatusWatchdog (Polls the STATUS register of the microcontroller any handles any errors that may arise) 32

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

libidp/clamp_control.cc 33
libidp/clamp_control.h 34
libidp/hal.cc 34
libidp/hal.h 35
libidp/libidp.h 36
libidp/line_following.cc 37
libidp/line_following.h 38
libidp/mission_supervisor.cc 39
libidp/mission_supervisor.h 40
libidp/navigation.cc 41
libidp/navigation.h 41
libidp/self_tests.cc 42
libidp/self_tests.h 43
libidp/status_watchdog.cc 44
libidp/status_watchdog.h 44

Chapter 5

Namespace Documentation

5.1 IDP Namespace Reference

Classes

- class ClampControl

  Manage the actuation of the clamp, as well as the detection and analysis of bobbins for their colour and badness.

- struct LineSensors

  Contains the LINE or NO_LINE status of each of the four IR sensors used for line following.

- class HardwareAbstractionLayer

  Provide a hardware agnostic interface to the required hardware functionality.

- class LineFollowing

  Maintain the robot position correctly with respect to the white line markers, during driving and manouvering.

- class MissionSupervisor

  Control the overall robot behaviour and objective fulfillment.

- class Navigation

  Find a route from one place to another on the board, and maintain an estimate of the current position.

- class SelfTests

  Execute a variety of functionality self tests.

- class StatusWatchdog

  Polls the STATUS register of the microcontroller any handles any errors that may arise.

Enumerations

- enum BobbinColour { BOBBIN_RED, BOBBIN_GREEN, BOBBIN_WHITE }

Bobbin colours.

- enum BobbinBadness { BOBBIN_GOOD, BOBBIN_BAD }

  Bobbin good or bad.

- enum LineSensorStatus { LINE, NO_LINE }

  Line sensor status, LINE or NO_LINE.

- enum LineFollowingStatus {

  ACTION_IN_PROGRESS, ACTION_COMPLETED, LEFT_TURN_FOUND, RIGHT_TURN_-FOUND,

  BOTH_TURNS_FOUND, LOST }

  Line following return status codes.

- enum NavigationStatus { NAVIGATION_ENROUTE, NAVIGATION_ARRIVED, NAVIGATION_LOST }

  Current navigation status.

- enum NavigationLocation { NAVIGATION_BOXES, NAVIGATION_RACK, NAVIGATION_-DELIVERY }

  Navigation's current position estimate.

Functions

- unsigned short int cap_correction (const unsigned short int correction)

  Cap a line following correction value to MAX_CORRECTION.

Variables

- const int MOTOR_MAX_SPEED = 127

  Highest allowable motor speed in either direction.

- const int MOTOR_RAMP_TIME = 16

  How fast to ramp the motors towards the desired speed.

- const double INTEGRAL_GAIN = 4.0

  Constant for integral control in line following.

- const short unsigned int MAX_CORRECTION = 127

  Maximum differential correction value before it gets capped.

- const unsigned int LOST_TIMEOUT = 50

  The number of loop iterations before we count as lost.

- const unsigned int EDGE_ERROR = 2

  How much an outer sensor seeing the edge of a line should add to the appropriate error.

5.1.1 Enumeration Type Documentation

5.1.1.1 enum IDP::BobbinBadness

Bobbin good or bad.

Enumerator:

BOBBIN_GOOD
BOBBIN_BAD

5.1.1.2 enum IDP::BobbinColour

Bobbin colours.

Enumerator:

BOBBIN_RED
BOBBIN_GREEN
BOBBIN_WHITE

5.1.1.3 enum IDP::LineFollowingStatus

Line following return status codes.

ACTION_IN_PROGRESS indicates that the requested action is still underway.

ACTION_COMPLETED indicates that the requested action has finished.

LEFT_TURN_FOUND, RIGHT_TURN_FOUND and BOTH_TURNS_FOUND indicate that possible turns have been found in the path.

LOST indicates that no line could be seen on any sensors and that this is unexpected.

Enumerator:

ACTION_IN_PROGRESS
ACTION_COMPLETED
LEFT_TURN_FOUND
RIGHT_TURN_FOUND
BOTH_TURNS_FOUND
LOST

5.1.1.4 enum IDP::LineSensorStatus

Line sensor status, LINE or NO_LINE.

Enumerator:

LINE
NO_LINE

5.1.1.5 enum IDP::NavigationLocation

Navigation's current position estimate.

Enumerator:

NAVIGATION_BOXES

NAVIGATION_RACK

NAVIGATION_DELIVERY

5.1.1.6 enum IDP::NavigationStatus

Current navigation status.

Enumerator:

NAVIGATION_ENROUTE

NAVIGATION_ARRIVED

NAVIGATION_LOST

5.1.2 Function Documentation

5.1.2.1 unsigned short int IDP::cap_correction (const unsigned short int *correction*)

Cap a line following correction value to MAX_CORRECTION.

Parameters

correction The existing correction value

Returns

The capped correction value

5.1.3 Variable Documentation

5.1.3.1 const unsigned int IDP::EDGE_ERROR = 2

How much an outer sensor seeing the edge of a line should add to the appropriate error.

5.1.3.2 const double IDP::INTEGRAL_GAIN = 4.0

Constant for integral control in line following.

5.1.3.3 const unsigned int IDP::LOST_TIMEOUT = 50

The number of loop iterations before we count as lost.

5.1.3.4 const short unsigned int IDP::MAX_CORRECTION = 127

Maximum differential correction value before it gets capped.

5.1.3.5 const int IDP::MOTOR_MAX_SPEED = 127

Highest allowable motor speed in either direction.

5.1.3.6 const int IDP::MOTOR_RAMP_TIME = 16

How fast to ramp the motors towards the desired speed.

Lower is faster.

Chapter 6

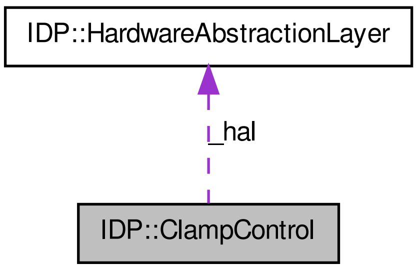
Class Documentation

6.1 IDP::ClampControl Class Reference

Manage the actuation of the clamp, as well as the detection and analysis of bobbins for their colour and badness.

`#include <clamp_control.h>`

Collaboration diagram for IDP::ClampControl:



Public Member Functions

- ClampControl (const HardwareAbstractionLayer ∗hal)

  Initialise the class, storing the const pointer to the HAL.

- void pick_up ()

  Pick up something using the clamp.

- void put_down ()

  Put something in the clamp down.

- const BobbinColour colour () const

  Check the bobbin colour.

- const BobbinBadness badness () const

  Check the bobbin badness.

6.1.1 Detailed Description

Manage the actuation of the clamp, as well as the detection and analysis of bobbins for their colour and badness.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 IDP::ClampControl::ClampControl (const HardwareAbstractionLayer $*$ *hal*)

Initialise the class, storing the const pointer to the HAL.

Parameters

hal A const pointer to an instance of the HAL

6.1.3 Member Function Documentation

6.1.3.1 const BobbinBadness IDP::ClampControl::badness ( ) const

Check the bobbin badness.

Returns

A BobbinBadness value to indicate current bobbin status

6.1.3.2 const BobbinColour IDP::ClampControl::colour ( ) const

Check the bobbin colour.

Returns

A BobbinColour value to indicate current bobbin colour

6.1.3.3 void IDP::ClampControl::pick_up ( )

Pick up something using the clamp.

6.1.3.4 void IDP::ClampControl::put_down ( )

Put something in the clamp down.

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

- libidp/clamp_control.h
- libidp/clamp_control.cc

6.2 IDP::HardwareAbstractionLayer Class Reference

Provide a hardware agnostic interface to the required hardware functionality.

```
#include <hal.h>
```

Public Member Functions

- HardwareAbstractionLayer (const int robot)

    Initialise the HAL class.

- void motors_forward (const unsigned short int speed) const

    Drive both motors forwards at a given speed.

- void motors_backward (const unsigned short int speed) const

    Drive both motors backwards at a given speed.

- void motor_left_forward (const unsigned short int speed) const

    Drive the left motor forward at the given speed.

- void motor_right_forward (const unsigned short int speed) const

    Drive the right motor forward at the given speed.

- void motor_left_backward (const unsigned short int speed) const

    Drive the left motor backward at the given speed.

- void motor_right_backward (const unsigned short int speed) const

    Drive the right motor backward at the given speed.

- void motors_turn_left (const unsigned short int speed) const

    Drive the motors to steer the robot to the left.

- void motors_turn_right (const unsigned short int speed) const

    Drive the motors to steer the robot to the right.

- void motors_stop () const

    Stop all motors.

- char status_register () const

    Read the status register and return it.

- void clear_status_register () const

  Read the status register, discarding its value.

- const LineSensors line_following_sensors () const

  Read the I/O port connected to the line following sensors, then return a struct with their current state.

- bool reset_switch () const

  Read the reset switch and return its status.

- bool grabber_switch () const

  Read the switch mounted on the grabber arm and return its status.

- unsigned short int colour_ldr () const

  Get the analogue reading from the LDR used to detect colour.

- unsigned short int bad_bobbin_ldr () const

  Get the analogue reading from the LDR used to detect the bad bobbin.

- void indication_LEDs (const bool led_0, const bool led_1, const bool led_2) const

  Set the bobbin colour indication LEDs.

- void colour_leds (const bool red, const bool green) const

  Turn on and off the LEDs used to light up the bobbin for colour detection.

- void bad_bobbin_led (const bool status) const

  Turn on and off the LED used to light up the top of the bobbin, for bad bobbin detection.

- void grabber_jaw (const bool status) const

  Turn the grabber jaw actuator on or off.

- void grabber_lift (const bool status) const

  Turn the grabber lift mechanism actuator on or off.

6.2.1 Detailed Description

Provide a hardware agnostic interface to the required hardware functionality.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 IDP::HardwareAbstractionLayer::HardwareAbstractionLayer (const int *robot* = `0`)

Initialise the HAL class.

Establishes the link to the robot.

6.2.3 Member Function Documentation

6.2.3.1 unsigned short int IDP::HardwareAbstractionLayer::bad_bobbin_ldr ( ) const

Get the analogue reading from the LDR used to detect the bad bobbin.

Returns

The analogue reading value

6.2.3.2 void IDP::HardwareAbstractionLayer::bad_bobbin_led (const bool *status*) const

Turn on and off the LED used to light up the top of the bobbin, for bad bobbin detection.

Parameters

status	Whether the LED should be on or off (true=on)

6.2.3.3 void IDP::HardwareAbstractionLayer::clear_status_register ( ) const

Read the status register, discarding its value.

6.2.3.4 unsigned short int IDP::HardwareAbstractionLayer::colour_ldr ( ) const

Get the analogue reading from the LDR used to detect colour.

Returns

The analogue reading value

6.2.3.5 void IDP::HardwareAbstractionLayer::colour_leds (const bool *red,* const bool *green*) const

Turn on and off the LEDs used to light up the bobbin for colour detection.

Parameters

red	Whether the red LED should be on or off (true=on)
green	Whether the green LED should be on or off (true=on)

6.2.3.6 void IDP::HardwareAbstractionLayer::grabber_jaw (const bool *status*) const

Turn the grabber jaw actuator on or off.

Parameters

status	Jaw actuator status (true=on)

6.2.3.7 void IDP::HardwareAbstractionLayer::grabber_lift (const bool *status*) const

Turn the grabber lift mechanism actuator on or off.

Parameters

status Lift actuator status (true=on)

6.2.3.8 bool IDP::HardwareAbstractionLayer::grabber_switch ( ) const

Read the switch mounted on the grabber arm and return its status.

Returns

The current value of the switch, true if pressed

6.2.3.9 void IDP::HardwareAbstractionLayer::indication_LEDs (const bool *led_0,* const bool *led_1,* const bool *led_2*) const

Set the bobbin colour indication LEDs.

Parameters

led_0	Whether LED0 should be on or off (true=on)
led_1	Whether LED1 should be on or off (true=on)
led_2	Whether LED2 should be on or off (true=on)

6.2.3.10 const LineSensors IDP::HardwareAbstractionLayer::line_following_sensors ( ) const

Read the I/O port connected to the line following sensors, then return a struct with their current state.

Returns

A LineSensors struct containing the current state of the sensors

6.2.3.11 void IDP::HardwareAbstractionLayer::motor_left_backward (const unsigned short int *speed*) const

Drive the left motor backward at the given speed.

Parameters

speed The speed at which to drive the motor

6.2.3.12 void IDP::HardwareAbstractionLayer::motor_left_forward (const unsigned short int *speed*) const

Drive the left motor forward at the given speed.

Parameters

speed The speed at which to drive the motor

6.2.3.13 void IDP::HardwareAbstractionLayer::motor_right_backward (const unsigned short int *speed*) const

Drive the right motor backward at the given speed.

Parameters

speed The speed at which to drive the motor

6.2.3.14 void IDP::HardwareAbstractionLayer::motor_right_forward (const unsigned short int *speed*) const

Drive the right motor forward at the given speed.

Parameters

speed The speed at which to drive the motor

6.2.3.15 void IDP::HardwareAbstractionLayer::motors_backward (const unsigned short int *speed*) const

Drive both motors backwards at a given speed.

Parameters

speed The speed to drive at, 0 to 127

6.2.3.16 void IDP::HardwareAbstractionLayer::motors_forward (const unsigned short int *speed*) const

Drive both motors forwards at a given speed.

Parameters

speed The speed to drive at, 0 to 127

6.2.3.17 void IDP::HardwareAbstractionLayer::motors_stop ( ) const

Stop all motors.

6.2.3.18 void IDP::HardwareAbstractionLayer::motors_turn_left (const unsigned short int *speed*) const

Drive the motors to steer the robot to the left.

Parameters

speed The speed to drive at, 0 to 127

6.2.3.19 void IDP::HardwareAbstractionLayer::motors_turn_right (const unsigned short int *speed*) const

Drive the motors to steer the robot to the right.

Parameters

speed The speed to drive at, 0 to 127

6.2.3.20 bool IDP::HardwareAbstractionLayer::reset_switch ( ) const

Read the reset switch and return its status.

Returns

The current value of the switch, true if pressed

6.2.3.21 char IDP::HardwareAbstractionLayer::status_register ( ) const

Read the status register and return it.

Returns

The STATUS register

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

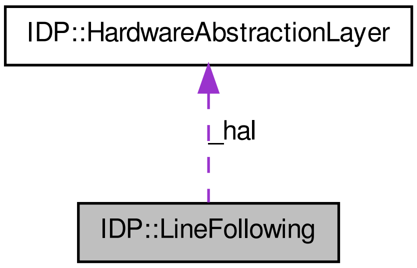
- libidp/hal.h
- libidp/hal.cc

6.3 IDP::LineFollowing Class Reference

Maintain the robot position correctly with respect to the white line markers, during driving and manouvering.

```
#include <line_following.h>
```

Collaboration diagram for IDP::LineFollowing:



Public Member Functions

- LineFollowing (const HardwareAbstractionLayer ∗hal)

  Construct the Line Follower.

- LineFollowingStatus follow_line (void)

  Read line sensors and correct motor movement to keep us going straight.

- LineFollowingStatus turn_left (void)

  Turn the robot left until the sensors encounter another line.

- LineFollowingStatus turn_right (void)

  Turn the robot right until the sensors detect another line.

- void set_speed (unsigned short int speed)

  Set the speed that motors will be driven at.

6.3.1 Detailed Description

Maintain the robot position correctly with respect to the white line markers, during driving and manouvering.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 IDP::LineFollowing::LineFollowing (const HardwareAbstractionLayer ∗ *hal*)

Construct the Line Follower.

6.3.3 Member Function Documentation

6.3.3.1 LineFollowingStatus IDP::LineFollowing::follow_line (void)

Read line sensors and correct motor movement to keep us going straight.

Returns

A LineFollowingStatus to indicate that either we are going fine, we are lost, or one or more possible turns were found.

6.3.3.2 void IDP::LineFollowing::set_speed (unsigned short int *speed*)

Set the speed that motors will be driven at.

Parameters

speed How fast to drive the motors, 0 to MOTOR_MAX_SPEED.

6.3.3.3 LineFollowingStatus IDP::LineFollowing::turn_left (void)

Turn the robot left until the sensors encounter another line.

6.3.3.4 LineFollowingStatus IDP::LineFollowing::turn_right (void)

Turn the robot right until the sensors detect another line.

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

- libidp/line_following.h
- libidp/line_following.cc

6.4 IDP::LineSensors Struct Reference

Contains the LINE or NO_LINE status of each of the four IR sensors used for line following.

`#include <hal.h>`

Public Attributes

- LineSensorStatus outer_left
- LineSensorStatus line_left
- LineSensorStatus line_right
- LineSensorStatus outer_right

6.4.1 Detailed Description

Contains the LINE or NO_LINE status of each of the four IR sensors used for line following.

6.4.2 Member Data Documentation

6.4.2.1 LineSensorStatus IDP::LineSensors::line_left

6.4.2.2 LineSensorStatus IDP::LineSensors::line_right

6.4.2.3 LineSensorStatus IDP::LineSensors::outer_left

6.4.2.4 LineSensorStatus IDP::LineSensors::outer_right

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

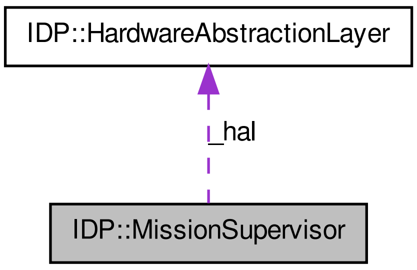
- libidp/hal.h

6.5 IDP::MissionSupervisor Class Reference

Control the overall robot behaviour and objective fulfillment.

`#include <mission_supervisor.h>`

Collaboration diagram for IDP::MissionSupervisor:



Public Member Functions

- MissionSupervisor (int robot)

  Construct the MissionSupervisor.

- void run_task ()

  Commence running the main task.

- void drive_forward ()

  Set both motors driving forwards.

- void drive_backward ()

Set both motors driving backwards.

- void stop ()

  Stop all motors.

- void test_line_sensor ()

  Attempt to read the line sensor status.

- void test_line_following ()

  Test line following on a straight line.

- const HardwareAbstractionLayer ∗ hal () const

  Const accessor for the HAL.

6.5.1 Detailed Description

Control the overall robot behaviour and objective fulfillment.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 IDP::MissionSupervisor::MissionSupervisor (int *robot* = 0)

Construct the MissionSupervisor.

Initialises a link to the specified robot number, or 0 if running embedded.

Parameters

robot Which robot to link to, or 0 if embedded

6.5.3 Member Function Documentation

6.5.3.1 void IDP::MissionSupervisor::drive_backward ( )

Set both motors driving backwards.

6.5.3.2 void IDP::MissionSupervisor::drive_forward ( )

Set both motors driving forwards.

6.5.3.3 const HardwareAbstractionLayer ∗ IDP::MissionSupervisor::hal ( ) const

Const accessor for the HAL.

6.5.3.4 void IDP::MissionSupervisor::run_task ( )

Commence running the main task.

6.5.3.5 void IDP::MissionSupervisor::stop ( )

Stop all motors.

6.5.3.6 void IDP::MissionSupervisor::test_line_following ( )

Test line following on a straight line.

6.5.3.7 void IDP::MissionSupervisor::test_line_sensor ( )

Attempt to read the line sensor status.

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

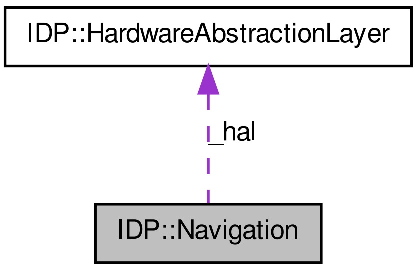
- libidp/mission_supervisor.h
- libidp/mission_supervisor.cc

6.6 IDP::Navigation Class Reference

Find a route from one place to another on the board, and maintain an estimate of the current position.

`#include <navigation.h>`

Collaboration diagram for IDP::Navigation:



Public Member Functions

- Navigation (const HardwareAbstractionLayer ∗hal)

  Initialise the class, storing the const pointer to the HAL.

- const NavigationStatus go (const NavigationLocation location)

  Go to a location.

6.6.1 Detailed Description

Find a route from one place to another on the board, and maintain an estimate of the current position.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 IDP::Navigation::Navigation (const HardwareAbstractionLayer ∗ *hal*)

Initialise the class, storing the const pointer to the HAL.

Parameters

hal A const pointer to an instance of the HAL

6.6.3 Member Function Documentation

6.6.3.1 const NavigationStatus IDP::Navigation::go (const NavigationLocation *location*)

Go to a location.

Returns

A navigation status code

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

- libidp/navigation.h
- libidp/navigation.cc

6.7 IDP::SelfTests Class Reference

Execute a variety of functionality self tests.

```
#include <self_tests.h>
```

Public Member Functions

- SelfTests (int robot)

  Constuct a SelfTest instance Completely seperate to mission supervisor and initialises own link to robot, with its own HAL instance.

- void drive_forward (void)

  Drive the robot forwards for a moment.

- void drive_backward (void)

  Drive the robot backwards for a moment.

- void stop (void)

  Stop all of the robot's motors.

- void turn_left (void)

  Drive motors in opposite directions to turn the robot left on the spot.

- void turn_right (void)

  Drive motors in opposite directions to turn the robot right on the spot.

- void steer_left (void)

  Drive forwards for a moment whilst reducing the speed of the left motor relative to the right to steer left.

- void steer_right (void)

  Drive forwards for a moment whilst reducing the speed of the right motor relative to the left to steer right.

- void line_sensors (void)

  Display the status (LINE or NO_LINE) of each of the four IR line following sensors.

- void microswitches (void)

  Display the state of each of the two microswitches.

- void LDRs (void)

  Display the current ADC read from the light dependent resistor.

- void actuators (void)

  Fire each of the actuators in turn.

- void line_following (void)

  Follow a line until further notice, without caring where we end up.

- void clamp_control (void)

  Use the actuators to pick up an object before placing it back down again.

- void bobbin_analyse (void)

  Analyse the colour of the bobbin that is currently being held in the clamp.

- void navigate (void)

  Select a source and destination and then navigate to the destination assuming we are starting at the source.

- void position (void)

  Drive slowly looking for an object in range for pickup, then position self ready to clamp said object.

- void status_LEDs (void)

  Turn on each of the status LEDs (used for indicating bobbin colour) in turn.

- void colour_sensor_LEDs (void)

  Turn on each of the coloured LEDs used for colour detection in turn.

- void badness_LED (void)

  Turn on the LED used for detecting bad bobbins.

6.7.1 Detailed Description

Execute a variety of functionality self tests.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 IDP::SelfTests::SelfTests (int *robot* = `0`)

Constuct a SelfTest instance Completely seperate to mission supervisor and initialises own link to robot, with its own HAL instance.

Parameters

robot	Which robot to link to, or 0 if embedded

6.7.3 Member Function Documentation

6.7.3.1 void IDP::SelfTests::actuators (void)

Fire each of the actuators in turn.

6.7.3.2 void IDP::SelfTests::badness_LED (void)

Turn on the LED used for detecting bad bobbins.

6.7.3.3 void IDP::SelfTests::bobbin_analyse (void)

Analyse the colour of the bobbin that is currently being held in the clamp.

6.7.3.4 void IDP::SelfTests::clamp_control (void)

Use the actuators to pick up an object before placing it back down again.

6.7.3.5 void IDP::SelfTests::colour_sensor_LEDs (void)

Turn on each of the coloured LEDs used for colour detection in turn.

6.7.3.6 void IDP::SelfTests::drive_backward (void)

Drive the robot backwards for a moment.

6.7.3.7 void IDP::SelfTests::drive_forward (void)

Drive the robot forwards for a moment.

6.7.3.8 void IDP::SelfTests::LDRs (void)

Display the current ADC read from the light dependent resistor.

6.7.3.9 void IDP::SelfTests::line_following (void)

Follow a line until further notice, without caring where we end up.

6.7.3.10 void IDP::SelfTests::line_sensors (void)

Display the status (LINE or NO_LINE) of each of the four IR line following sensors.

6.7.3.11 void IDP::SelfTests::microswitches (void)

Display the state of each of the two microswitches.

6.7.3.12 void IDP::SelfTests::navigate (void)

Select a source and destination and then navigate to the destination assuming we are starting at the source.

6.7.3.13 void IDP::SelfTests::position (void)

Drive slowly looking for an object in range for pickup, then position self ready to clamp said object.

6.7.3.14 void IDP::SelfTests::status_LEDs (void)

Turn on each of the status LEDs (used for indicating bobbin colour) in turn.

6.7.3.15 void IDP::SelfTests::steer_left (void)

Drive forwards for a moment whilst reducing the speed of the left motor relative to the right to steer left.

6.7.3.16 void IDP::SelfTests::steer_right (void)

Drive forwards for a moment whilst reducing the speed of the right motor relative to the left to steer right.

6.7.3.17 void IDP::SelfTests::stop (void)

Stop all of the robot's motors.

6.7.3.18 void IDP::SelfTests::turn_left (void)

Drive motors in opposite directions to turn the robot left on the spot.

6.7.3.19 void IDP::SelfTests::turn_right (void)

Drive motors in opposite directions to turn the robot right on the spot.

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

- libidp/self_tests.h

- libidp/self_tests.cc

6.8 IDP::StatusWatchdog Class Reference

Polls the STATUS register of the microcontroller any handles any errors that may arise.

```
#include <status_watchdog.h>
```

Public Member Functions

- const int check () const

  Read the STATUS register of the microcontroller and return the value.

6.8.1 Detailed Description

Polls the STATUS register of the microcontroller any handles any errors that may arise.

6.8.2 Member Function Documentation

6.8.2.1 const int IDP::StatusWatchdog::check ( ) const

Read the STATUS register of the microcontroller and return the value.

Returns

The error encountered, if any

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

- libidp/status_watchdog.h
- libidp/status_watchdog.cc

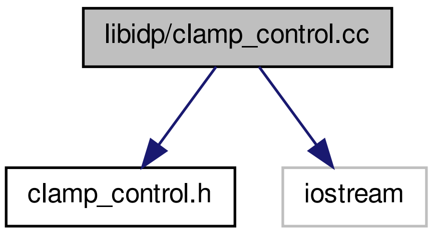
Chapter 7

File Documentation

7.1 libidp/clamp_control.cc File Reference

```
#include "clamp_control.h"
#include <iostream>
```

Include dependency graph for clamp_control.cc:

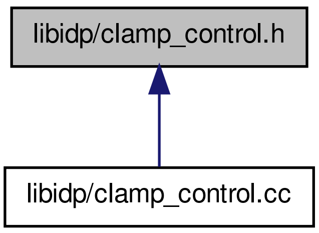


Namespaces

- namespace IDP

7.2 libidp/clamp_control.h File Reference

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



Classes

- class IDP::ClampControl

  Manage the actuation of the clamp, as well as the detection and analysis of bobbins for their colour and badness.

Namespaces

- namespace IDP

Enumerations

- enum IDP::BobbinColour { IDP::BOBBIN_RED, IDP::BOBBIN_GREEN, IDP::BOBBIN_WHITE }

  Bobbin colours.

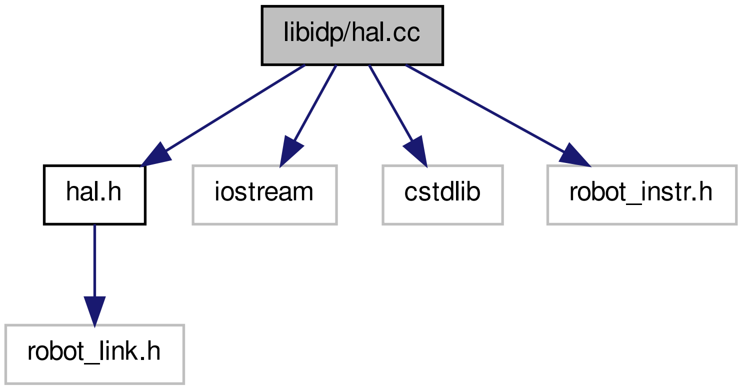
- enum IDP::BobbinBadness { IDP::BOBBIN_GOOD, IDP::BOBBIN_BAD }

  Bobbin good or bad.

7.3 libidp/hal.cc File Reference

```
#include "hal.h"
#include <iostream>
#include <cstdlib>
#include <robot_instr.h>
```

Include dependency graph for hal.cc:



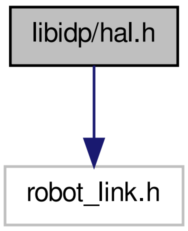
Namespaces

- namespace IDP

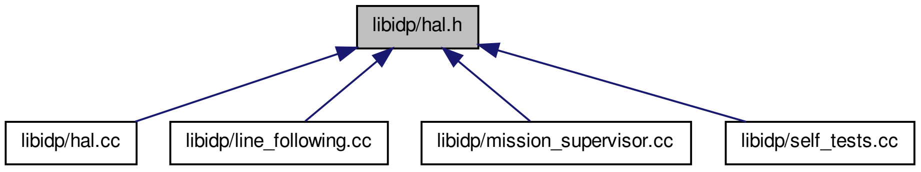
7.4 libidp/hal.h File Reference

```
#include <robot_link.h>
```

Include dependency graph for hal.h:



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



Classes

- struct IDP::LineSensors

  Contains the LINE or NO_LINE status of each of the four IR sensors used for line following.

- class IDP::HardwareAbstractionLayer

  Provide a hardware agnostic interface to the required hardware functionality.

Namespaces

- namespace IDP

Enumerations

- enum IDP::LineSensorStatus { IDP::LINE, IDP::NO_LINE }

  Line sensor status, LINE or NO_LINE.

Variables

- const int IDP::MOTOR_MAX_SPEED = 127

  Highest allowable motor speed in either direction.

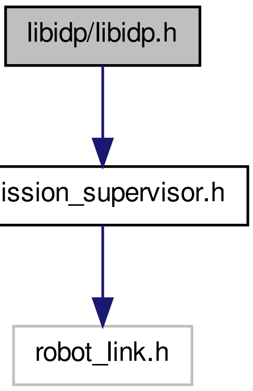
- const int IDP::MOTOR_RAMP_TIME = 16

  How fast to ramp the motors towards the desired speed.

7.5 libidp/libidp.h File Reference

```
#include "mission_supervisor.h"
```

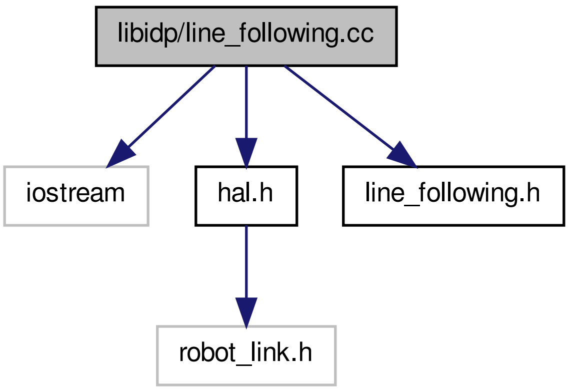
Include dependency graph for libidp.h:



7.6 libidp/line_following.cc File Reference

```
#include <iostream>
#include "hal.h"
#include "line_following.h"
```

Include dependency graph for line_following.cc:



Namespaces

- namespace IDP

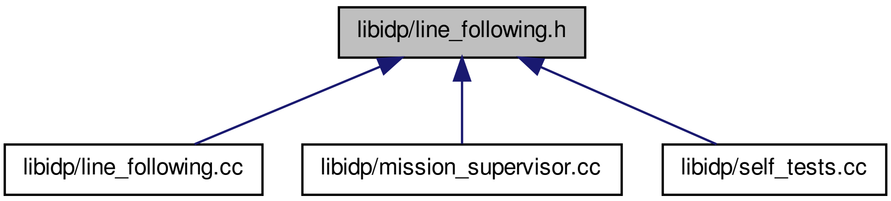
Functions

- unsigned short int IDP::cap_correction (const unsigned short int correction)

    Cap a line following correction value to MAX_CORRECTION.

7.7 libidp/line_following.h File Reference

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



Classes

- class IDP::LineFollowing

    Maintain the robot position correctly with respect to the white line markers, during driving and manouvering.

Namespaces

- namespace IDP

Enumerations

- enum IDP::LineFollowingStatus {

  IDP::ACTION_IN_PROGRESS, IDP::ACTION_COMPLETED, IDP::LEFT_TURN_FOUND, IDP::RIGHT_TURN_FOUND,

  IDP::BOTH_TURNS_FOUND, IDP::LOST }

    Line following return status codes.

Functions

- unsigned short int IDP::cap_correction (const unsigned short int correction)

  Cap a line following correction value to MAX_CORRECTION.

Variables

- const double IDP::INTEGRAL_GAIN = 4.0

  Constant for integral control in line following.

- const short unsigned int IDP::MAX_CORRECTION = 127

  Maximum differential correction value before it gets capped.

- const unsigned int IDP::LOST_TIMEOUT = 50

  The number of loop iterations before we count as lost.

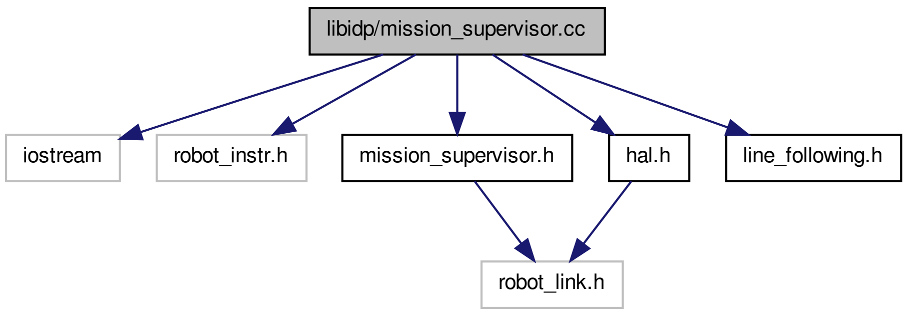
- const unsigned int IDP::EDGE_ERROR = 2

  How much an outer sensor seeing the edge of a line should add to the appropriate error.

7.8 libidp/mission_supervisor.cc File Reference

```
#include <iostream>
#include <robot_instr.h>
#include "mission_supervisor.h"
#include "hal.h"
#include "line_following.h"
```

Include dependency graph for mission_supervisor.cc:



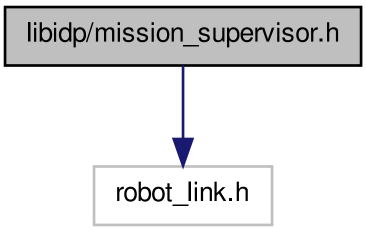
Namespaces

- namespace IDP

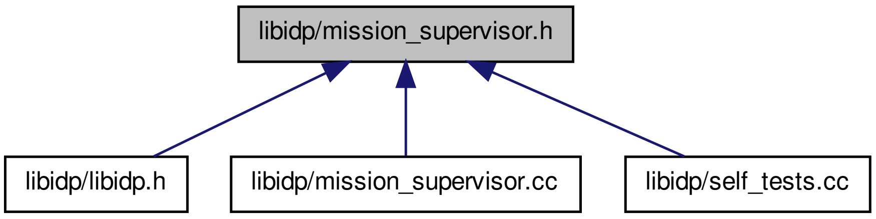
7.9 libidp/mission_supervisor.h File Reference

```
#include <robot_link.h>
```

Include dependency graph for mission_supervisor.h:



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



Classes

- class IDP::MissionSupervisor

  Control the overall robot behaviour and objective fulfillment.

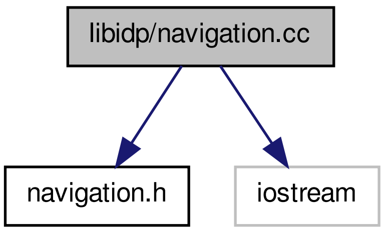
Namespaces

- namespace IDP

7.10 libidp/navigation.cc File Reference

```
#include "navigation.h"
#include <iostream>
```

Include dependency graph for navigation.cc:

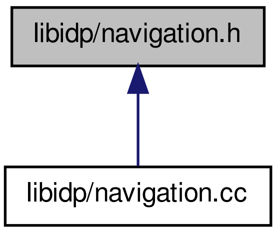


Namespaces

- namespace IDP

7.11 libidp/navigation.h File Reference

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



Classes

- class IDP::Navigation

  Find a route from one place to another on the board, and maintain an estimate of the current position.

Namespaces

- namespace IDP

Enumerations

- enum IDP::NavigationStatus { IDP::NAVIGATION_ENROUTE, IDP::NAVIGATION_ARRIVED, IDP::NAVIGATION_LOST }

  Current navigation status.

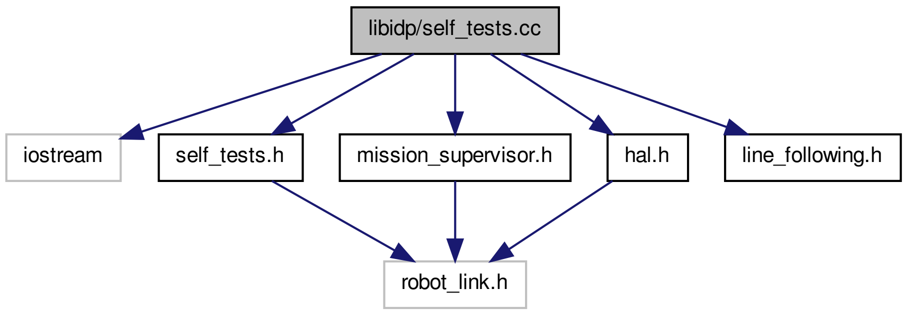
- enum IDP::NavigationLocation { IDP::NAVIGATION_BOXES, IDP::NAVIGATION_RACK, IDP::NAVIGATION_DELIVERY }

  Navigation's current position estimate.

7.12 libidp/self_tests.cc File Reference

```
#include <iostream>
#include "self_tests.h"
#include "mission_supervisor.h"
#include "hal.h"
#include "line_following.h"
```

Include dependency graph for self_tests.cc:



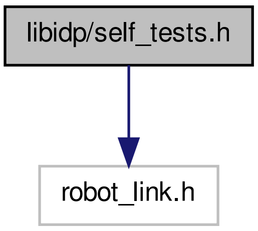
Namespaces

- namespace IDP

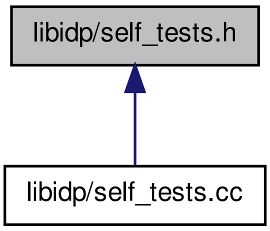
7.13 libidp/self_tests.h File Reference

```
#include <robot_link.h>
```

Include dependency graph for self_tests.h:



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



Classes

- class IDP::SelfTests

  Execute a variety of functionality self tests.

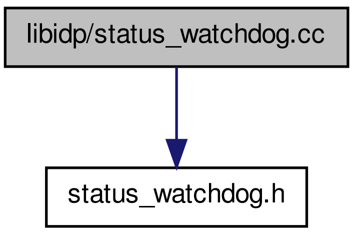
Namespaces

- namespace IDP

7.14 libidp/status_watchdog.cc File Reference

```
#include "status_watchdog.h"
```

Include dependency graph for status_watchdog.cc:

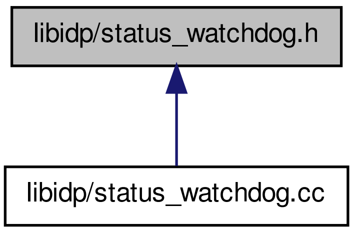


Namespaces

- namespace IDP

7.15 libidp/status_watchdog.h File Reference

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



Classes

- class IDP::StatusWatchdog

  Polls the STATUS register of the microcontroller any handles any errors that may arise.

Namespaces

- namespace IDP

Index

ACTION_COMPLETED
    IDP, 11
ACTION_IN_PROGRESS
    IDP, 11
actuators
    IDP::SelfTests, 30

bad_bobbin_ldr
    IDP::HardwareAbstractionLayer, 19
bad_bobbin_led
    IDP::HardwareAbstractionLayer, 19
badness
    IDP::ClampControl, 16
badness_LED
    IDP::SelfTests, 30
BOBBIN_BAD
    IDP, 11
BOBBIN_GOOD
    IDP, 11
BOBBIN_GREEN
    IDP, 11
BOBBIN_RED
    IDP, 11
BOBBIN_WHITE
    IDP, 11
bobbin_analyse
    IDP::SelfTests, 30
BobbinBadness
    IDP, 11
BobbinColour
    IDP, 11
BOTH_TURNS_FOUND
    IDP, 11

cap_correction
    IDP, 12
check
    IDP::StatusWatchdog, 32
clamp_control
    IDP::SelfTests, 30
ClampControl
    IDP::ClampControl, 16
clear_status_register
    IDP::HardwareAbstractionLayer, 19
colour
    IDP::ClampControl, 16
colour_ldr
    IDP::HardwareAbstractionLayer, 19
colour_leds
    IDP::HardwareAbstractionLayer, 19
colour_sensor_LEDs
    IDP::SelfTests, 30

drive_backward
    IDP::MissionSupervisor, 26
    IDP::SelfTests, 30
drive_forward
    IDP::MissionSupervisor, 26
    IDP::SelfTests, 30

EDGE_ERROR
    IDP, 12

follow_line
    IDP::LineFollowing, 24

go
    IDP::Navigation, 28
grabber_jaw
    IDP::HardwareAbstractionLayer, 19
grabber_lift
    IDP::HardwareAbstractionLayer, 19
grabber_switch
    IDP::HardwareAbstractionLayer, 20

hal
    IDP::MissionSupervisor, 26
HardwareAbstractionLayer
    IDP::HardwareAbstractionLayer, 18

IDP, 9
    ACTION_COMPLETED, 11
    ACTION_IN_PROGRESS, 11
    BOBBIN_BAD, 11
    BOBBIN_GOOD, 11
    BOBBIN_GREEN, 11
    BOBBIN_RED, 11
    BOBBIN_WHITE, 11
    BobbinBadness, 11
    BobbinColour, 11
    BOTH_TURNS_FOUND, 11

cap_correction, 12
EDGE_ERROR, 12
INTEGRAL_GAIN, 12
LEFT_TURN_FOUND, 11
LINE, 11
LineFollowingStatus, 11
LineSensorStatus, 11
LOST, 11
LOST_TIMEOUT, 12
MAX_CORRECTION, 12
MOTOR_MAX_SPEED, 13
MOTOR_RAMP_TIME, 13
NAVIGATION_ARRIVED, 12
NAVIGATION_BOXES, 12
NAVIGATION_DELIVERY, 12
NAVIGATION_ENROUTE, 12
NAVIGATION_LOST, 12
NAVIGATION_RACK, 12
NavigationLocation, 11
NavigationStatus, 12
NO_LINE, 11
RIGHT_TURN_FOUND, 11
IDP::ClampControl, 15
badness, 16
ClampControl, 16
colour, 16
pick_up, 16
put_down, 16
IDP::HardwareAbstractionLayer, 17
bad_bobbin_ldr, 19
bad_bobbin_led, 19
clear_status_register, 19
colour_ldr, 19
colour_leds, 19
grabber_jaw, 19
grabber_lift, 19
grabber_switch, 20
HardwareAbstractionLayer, 18
indication_LEDs, 20
line_following_sensors, 20
motor_left_backward, 20
motor_left_forward, 20
motor_right_backward, 20
motor_right_forward, 21
motors_backward, 21
motors_forward, 21
motors_stop, 21
motors_turn_left, 21
motors_turn_right, 21
reset_switch, 22
status_register, 22
IDP::LineFollowing, 22
follow_line, 24
LineFollowing, 23
set_speed, 24
turn_left, 24
turn_right, 24
IDP::LineSensors, 24
line_left, 25
line_right, 25
outer_left, 25
outer_right, 25
IDP::MissionSupervisor, 25
drive_backward, 26
drive_forward, 26
hal, 26
MissionSupervisor, 26
run_task, 26
stop, 26
test_line_following, 27
test_line_sensor, 27
IDP::Navigation, 27
go, 28
Navigation, 28
IDP::SelfTests, 28
actuators, 30
badness_LED, 30
bobbin_analyse, 30
clamp_control, 30
colour_sensor_LEDs, 30
drive_backward, 30
drive_forward, 30
LDRs, 30
line_following, 30
line_sensors, 31
microswitches, 31
navigate, 31
position, 31
SelfTests, 30
status_LEDs, 31
steer_left, 31
steer_right, 31
stop, 31
turn_left, 31
turn_right, 31
IDP::StatusWatchdog, 32
check, 32
indication_LEDs
IDP::HardwareAbstractionLayer, 20
INTEGRAL_GAIN
IDP, 12

LDRs
IDP::SelfTests, 30
LEFT_TURN_FOUND
IDP, 11
libidp/clamp_control.cc, 33
libidp/clamp_control.h, 34

libidp/hal.cc, 34
libidp/hal.h, 35
libidp/libidp.h, 36
libidp/line_following.cc, 37
libidp/line_following.h, 38
libidp/mission_supervisor.cc, 39
libidp/mission_supervisor.h, 40
libidp/navigation.cc, 41
libidp/navigation.h, 41
libidp/self_tests.cc, 42
libidp/self_tests.h, 43
libidp/status_watchdog.cc, 44
libidp/status_watchdog.h, 44
LINE
    IDP, 11
line_following
    IDP::SelfTests, 30
line_following_sensors
    IDP::HardwareAbstractionLayer, 20
line_left
    IDP::LineSensors, 25
line_right
    IDP::LineSensors, 25
line_sensors
    IDP::SelfTests, 31
LineFollowing
    IDP::LineFollowing, 23
LineFollowingStatus
    IDP, 11
LineSensorStatus
    IDP, 11
LOST
    IDP, 11
LOST_TIMEOUT
    IDP, 12

MAX_CORRECTION
    IDP, 12
microswitches
    IDP::SelfTests, 31
MissionSupervisor
    IDP::MissionSupervisor, 26
motor_left_backward
    IDP::HardwareAbstractionLayer, 20
motor_left_forward
    IDP::HardwareAbstractionLayer, 20
MOTOR_MAX_SPEED
    IDP, 13
MOTOR_RAMP_TIME
    IDP, 13
motor_right_backward
    IDP::HardwareAbstractionLayer, 20
motor_right_forward
    IDP::HardwareAbstractionLayer, 21
motors_backward
    IDP::HardwareAbstractionLayer, 21
motors_forward
    IDP::HardwareAbstractionLayer, 21
motors_stop
    IDP::HardwareAbstractionLayer, 21
motors_turn_left
    IDP::HardwareAbstractionLayer, 21
motors_turn_right
    IDP::HardwareAbstractionLayer, 21

navigate
    IDP::SelfTests, 31
Navigation
    IDP::Navigation, 28
NAVIGATION_ARRIVED
    IDP, 12
NAVIGATION_BOXES
    IDP, 12
NAVIGATION_DELIVERY
    IDP, 12
NAVIGATION_ENROUTE
    IDP, 12
NAVIGATION_LOST
    IDP, 12
NAVIGATION_RACK
    IDP, 12
NavigationLocation
    IDP, 11
NavigationStatus
    IDP, 12
NO_LINE
    IDP, 11

outer_left
    IDP::LineSensors, 25
outer_right
    IDP::LineSensors, 25

pick_up
    IDP::ClampControl, 16
position
    IDP::SelfTests, 31
put_down
    IDP::ClampControl, 16

reset_switch
    IDP::HardwareAbstractionLayer, 22
RIGHT_TURN_FOUND
    IDP, 11
run_task
    IDP::MissionSupervisor, 26

SelfTests
    IDP::SelfTests, 30

set_speed
    IDP::LineFollowing, 24
status_LEDs
    IDP::SelfTests, 31
status_register
    IDP::HardwareAbstractionLayer, 22
steer_left
    IDP::SelfTests, 31
steer_right
    IDP::SelfTests, 31
stop
    IDP::MissionSupervisor, 26
    IDP::SelfTests, 31

test_line_following
    IDP::MissionSupervisor, 27
test_line_sensor
    IDP::MissionSupervisor, 27
turn_left
    IDP::LineFollowing, 24
    IDP::SelfTests, 31
turn_right
    IDP::LineFollowing, 24
    IDP::SelfTests, 31