

IDPL108

1.0

Generated by Doxygen 1.7.1

Tue Feb 8 2011 02:45:35

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	12
5.1.1.1	BobbinBadness	12
5.1.1.2	BobbinColour	12
5.1.1.3	LineFollowingLineStatus	12
5.1.1.4	LineFollowingStatus	12
5.1.1.5	LineFollowingTurnDirection	13
5.1.1.6	LineSensorStatus	13
5.1.1.7	NavigationCachedJunction	13
5.1.1.8	NavigationDirection	14
5.1.1.9	NavigationLocation	14
5.1.1.10	NavigationNode	14
5.1.1.11	NavigationStatus	15
5.1.1.12	NavigationTurn	15
5.1.2	Function Documentation	15
5.1.2.1	cap_correction	15
5.1.3	Variable Documentation	15
5.1.3.1	EDGE_ERROR	15

5.1.3.2	INTEGRAL_GAIN	16
5.1.3.3	LOST_TIMEOUT	16
5.1.3.4	MAX_CORRECTION	16
5.1.3.5	MOTOR_MAX_SPEED	16
5.1.3.6	MOTOR_RAMP_TIME	16
5.1.3.7	NAVIGATION_LOCATION_LOOKUP	16
5.1.3.8	NAVIGATION_NODE_TURNS	16
5.1.3.9	NAVIGATION_ROUTE_MAP	17
5.1.3.10	NAVIGATION_TURN_MAP	17
6	Class Documentation	19
6.1	IDP::ClampControl Class Reference	19
6.1.1	Detailed Description	20
6.1.2	Constructor & Destructor Documentation	20
6.1.2.1	ClampControl	20
6.1.3	Member Function Documentation	20
6.1.3.1	badness	20
6.1.3.2	colour	20
6.1.3.3	pick_up	21
6.1.3.4	put_down	21
6.2	IDP::HardwareAbstractionLayer Class Reference	21
6.2.1	Detailed Description	22
6.2.2	Constructor & Destructor Documentation	23
6.2.2.1	HardwareAbstractionLayer	23
6.2.3	Member Function Documentation	23
6.2.3.1	bad_bobbin_ldr	23
6.2.3.2	bad_bobbin_led	23
6.2.3.3	clear_status_register	23
6.2.3.4	colour_ldr	23
6.2.3.5	colour_leds	23
6.2.3.6	grabber_jaw	24
6.2.3.7	grabber_lift	24
6.2.3.8	grabber_switch	24
6.2.3.9	indication_LEDs	24
6.2.3.10	line_following_sensors	25
6.2.3.11	motor_left_backward	25
6.2.3.12	motor_left_forward	25

6.2.3.13	motor_right_backward	25
6.2.3.14	motor_right_forward	25
6.2.3.15	motors_backward	26
6.2.3.16	motors_forward	26
6.2.3.17	motors_stop	26
6.2.3.18	motors_turn_left	26
6.2.3.19	motors_turn_right	26
6.2.3.20	reset_switch	27
6.2.3.21	status_register	27
6.3	IDP::LineFollowing Class Reference	27
6.3.1	Detailed Description	28
6.3.2	Constructor & Destructor Documentation	28
6.3.2.1	LineFollowing	28
6.3.3	Member Function Documentation	28
6.3.3.1	follow_line	28
6.3.3.2	junction_status	29
6.3.3.3	set_speed	29
6.3.3.4	turn_around_ccw	29
6.3.3.5	turn_around_cw	29
6.3.3.6	turn_left	29
6.3.3.7	turn_right	30
6.4	IDP::LineSensors Struct Reference	30
6.4.1	Detailed Description	30
6.4.2	Member Data Documentation	30
6.4.2.1	line_left	30
6.4.2.2	line_right	30
6.4.2.3	outer_left	30
6.4.2.4	outer_right	31
6.5	IDP::MissionSupervisor Class Reference	31
6.5.1	Detailed Description	32
6.5.2	Constructor & Destructor Documentation	32
6.5.2.1	MissionSupervisor	32
6.5.2.2	~MissionSupervisor	32
6.5.3	Member Function Documentation	32
6.5.3.1	drive_backward	32
6.5.3.2	drive_forward	32

6.5.3.3	hal	33
6.5.3.4	run_task	33
6.5.3.5	stop	33
6.5.3.6	test_line_following	33
6.5.3.7	test_line_sensor	33
6.5.3.8	test_navigation	33
6.6	IDP::Navigation Class Reference	33
6.6.1	Detailed Description	34
6.6.2	Constructor & Destructor Documentation	35
6.6.2.1	Navigation	35
6.6.2.2	~Navigation	35
6.6.3	Member Function Documentation	35
6.6.3.1	go	35
6.6.3.2	go_node	35
6.7	IDP::SelfTests Class Reference	36
6.7.1	Detailed Description	37
6.7.2	Constructor & Destructor Documentation	37
6.7.2.1	SelfTests	37
6.7.3	Member Function Documentation	37
6.7.3.1	actuators	37
6.7.3.2	badness_LED	37
6.7.3.3	bobbin_analyse	38
6.7.3.4	clamp_control	38
6.7.3.5	colour_sensor_LEDs	38
6.7.3.6	drive_backward	38
6.7.3.7	drive_forward	38
6.7.3.8	LDRs	38
6.7.3.9	line_following	38
6.7.3.10	line_sensors	38
6.7.3.11	microswitches	38
6.7.3.12	navigate	39
6.7.3.13	position	39
6.7.3.14	status_LEDs	39
6.7.3.15	steer_left	39
6.7.3.16	steer_right	39
6.7.3.17	stop	39

6.7.3.18	turn_left	39
6.7.3.19	turn_right	39
6.8	IDP::StatusWatchdog Class Reference	40
6.8.1	Detailed Description	40
6.8.2	Member Function Documentation	40
6.8.2.1	check	40
7	File Documentation	41
7.1	src/libidp/clamp_control.cc File Reference	41
7.1.1	Define Documentation	42
7.1.1.1	DEBUG_ENABLED	42
7.1.1.2	ERROR_ENABLED	42
7.1.1.3	INFO_ENABLED	42
7.1.1.4	MODULE_NAME	42
7.1.1.5	TRACE_ENABLED	42
7.2	src/libidp/clamp_control.h File Reference	42
7.3	src/libidp/debug.h File Reference	43
7.4	src/libidp/hal.cc File Reference	44
7.4.1	Define Documentation	44
7.4.1.1	DEBUG_ENABLED	44
7.4.1.2	ERROR_ENABLED	45
7.4.1.3	INFO_ENABLED	45
7.4.1.4	MODULE_NAME	45
7.4.1.5	TRACE_ENABLED	45
7.4.1.6	UNUSED	45
7.5	src/libidp/hal.h File Reference	45
7.6	src/libidp/libidp.h File Reference	46
7.7	src/libidp/line_following.cc File Reference	47
7.7.1	Define Documentation	48
7.7.1.1	DEBUG_ENABLED	48
7.7.1.2	ERROR_ENABLED	48
7.7.1.3	INFO_ENABLED	48
7.7.1.4	MODULE_NAME	48
7.7.1.5	TRACE_ENABLED	48
7.8	src/libidp/line_following.h File Reference	49
7.9	src/libidp/mission_supervisor.cc File Reference	50
7.9.1	Define Documentation	51

7.9.1.1	DEBUG_ENABLED	51
7.9.1.2	ERROR_ENABLED	51
7.9.1.3	INFO_ENABLED	51
7.9.1.4	MODULE_NAME	51
7.9.1.5	TRACE_ENABLED	51
7.10	src/libidp/mission_supervisor.h File Reference	51
7.11	src/libidp/navigation.cc File Reference	52
7.11.1	Define Documentation	54
7.11.1.1	DEBUG_ENABLED	54
7.11.1.2	ERROR_ENABLED	54
7.11.1.3	INFO_ENABLED	54
7.11.1.4	MODULE_NAME	54
7.11.1.5	TRACE_ENABLED	54
7.11.1.6	UNUSED	54
7.12	src/libidp/navigation.h File Reference	54
7.13	src/libidp/self_tests.cc File Reference	56
7.13.1	Define Documentation	56
7.13.1.1	DEBUG_ENABLED	56
7.13.1.2	ERROR_ENABLED	56
7.13.1.3	INFO_ENABLED	57
7.13.1.4	MODULE_NAME	57
7.13.1.5	TRACE_ENABLED	57
7.14	src/libidp/self_tests.h File Reference	57
7.15	src/libidp/status_watchdog.cc File Reference	58
7.15.1	Define Documentation	59
7.15.1.1	DEBUG_ENABLED	59
7.15.1.2	ERROR_ENABLED	59
7.15.1.3	INFO_ENABLED	59
7.15.1.4	MODULE_NAME	59
7.15.1.5	TRACE_ENABLED	59
7.16	src/libidp/status_watchdog.h File Reference	59

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)	19
IDP::HardwareAbstractionLayer (Provide a hardware agnostic interface to the required hardware functionality)	21
IDP::LineFollowing (Maintain the robot position correctly with respect to the white line markers, during driving and manouvering)	27
IDP::LineSensors (Contains the LINE or NO_LINE status of each of the four IR sensors used for line following)	30
IDP::MissionSupervisor (Control the overall robot behaviour and objective fulfillment)	31
IDP::Navigation (Find a route from one place to another on the board, and maintain an estimate of the current position)	33
IDP::SelfTests (Execute a variety of functionality self tests)	36
IDP::StatusWatchdog (Polls the STATUS register of the microcontroller any handles any errors that may arise)	40

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/libidp/clamp_control.cc	41
src/libidp/clamp_control.h	42
src/libidp/debug.h	43
src/libidp/hal.cc	44
src/libidp/hal.h	45
src/libidp/libidp.h	46
src/libidp/line_following.cc	47
src/libidp/line_following.h	49
src/libidp/mission_supervisor.cc	50
src/libidp/mission_supervisor.h	51
src/libidp/navigation.cc	52
src/libidp/navigation.h	54
src/libidp/self_tests.cc	56
src/libidp/self_tests.h	57
src/libidp/status_watchdog.cc	58
src/libidp/status_watchdog.h	59

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`, `NO_TURNS_FOUND` }

Line following return status codes.

- enum `LineFollowingTurnDirection` {
`TURN_LEFT`, `TURN_RIGHT`, `TURN_AROUND_CW`, `TURN_AROUND_CCW`,
`MAX_TURN_DIRECTION` }

Possible turn directions.

- enum `LineFollowingLineStatus` { `ON_LINE`, `LOST_LINE`, `OTHER`, `MAX_LINE_STATUS` }

Possible line statuses, used internally.

- enum `NavigationStatus` { `NAVIGATION_ENROUTE`, `NAVIGATION_ARRIVED`,
`NAVIGATION_LOST`, `MAX_STATUS` }

Current navigation status.

- enum `NavigationLocation` { `NAVIGATION_BOXES`, `NAVIGATION_RACK`, `NAVIGATION_DELIVERY`, `MAX_LOCATION` }

Possible locations for navigation to be asked to go to.

- enum `NavigationDirection` { `NAVIGATION_CLOCKWISE`, `NAVIGATION_ANTICLOCKWISE`,
`MAX_DIRECTION` }

Directions around the circuit.

- enum `NavigationNode` {
`NODE1`, `NODE2`, `NODE3`, `NODE4`,
`NODE5`, `NODE6`, `NODE7`, `NODE8`,
`NODE9`, `NODE10`, `NODE11`, `MAX_NODE` }

Navigation nodes, numbered clockwise from the bottom right corner of the table.

- enum `NavigationTurn` {
`STRAIGHT`, `LEFT`, `RIGHT`, `BOTH`,
`LEFT_AND_STRAIGHT`, `RIGHT_AND_STRAIGHT`, `BOTH_AND_STRAIGHT`, `END_OF_LINE`,
`MAX_TURNS` }

Possible turns at a node.

- enum `NavigationCachedJunction` {
 `NO_CACHE`, `LEFT_TURN`, `RIGHT_TURN`, `BOTH_TURNS`,
 `NO_TURNS` }

Cached junction information for use while driving over a junction or executing a turn.

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.
- const `NavigationTurn` `NAVIGATION_NODE_TURNS` [`MAX_DIRECTION`][`MAX_NODE`]
The turns at each node.
- const `NavigationTurn` `NAVIGATION_TURN_MAP` [`MAX_DIRECTION`][`MAX_NODE`]
Turns that should be taken at each node in each direction.
- const `NavigationNode` `NAVIGATION_LOCATION_LOOKUP` [`MAX_LOCATION`][2]
The lookup table of `NavigationLocations` to a pair of `NavigationNodes` indicating the start and end node (with implied direction).
- const `NavigationNode` `NAVIGATION_ROUTE_MAP` [`MAX_DIRECTION`][`MAX_NODE`]
The route to take, node by node.

5.1.1 Enumeration Type Documentation

5.1.1.1 enum IDP::BobbinBadness

Bobbin good or bad.

Enumerator:

BOBBIN_GOOD

BOBBIN_BAD

Definition at line 30 of file clamp_control.h.

5.1.1.2 enum IDP::BobbinColour

Bobbin colours.

Enumerator:

BOBBIN_RED

BOBBIN_GREEN

BOBBIN_WHITE

Definition at line 21 of file clamp_control.h.

5.1.1.3 enum IDP::LineFollowingLineStatus

Possible line statuses, used internally.

Enumerator:

ON_LINE

LOST_LINE

OTHER

MAX_LINE_STATUS

Definition at line 76 of file line_following.h.

5.1.1.4 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
NO_TURNS_FOUND

Definition at line 52 of file line_following.h.

5.1.1.5 enum IDP::LineFollowingTurnDirection

Possible turn directions.

Enumerator:

TURN_LEFT
TURN_RIGHT
TURN_AROUND_CW
TURN_AROUND_CCW
MAX_TURN_DIRECTION

Definition at line 65 of file line_following.h.

5.1.1.6 enum IDP::LineSensorStatus

Line sensor status, LINE or NO_LINE.

Enumerator:

LINE
NO_LINE

Definition at line 32 of file hal.h.

5.1.1.7 enum IDP::NavigationCachedJunction

Cached junction information for use while driving over a junction or executing a turn.

Enumerator:

NO_CACHE
LEFT_TURN
RIGHT_TURN
BOTH_TURNS
NO_TURNS

Definition at line 60 of file navigation.h.

5.1.1.8 enum IDP::NavigationDirection

Directions around the circuit.

Enumerator:

NAVIGATION_CLOCKWISE
NAVIGATION_ANTICLOCKWISE
MAX_DIRECTION

Definition at line 35 of file navigation.h.

5.1.1.9 enum IDP::NavigationLocation

Possible locations for navigation to be asked to go to.

Enumerator:

NAVIGATION_BOXES
NAVIGATION_RACK
NAVIGATION_DELIVERY
MAX_LOCATION

Definition at line 28 of file navigation.h.

5.1.1.10 enum IDP::NavigationNode

[Navigation](#) nodes, numbered clockwise from the bottom right corner of the table.

Enumerator:

NODE1
NODE2
NODE3
NODE4
NODE5
NODE6
NODE7
NODE8
NODE9
NODE10
NODE11
MAX_NODE

Definition at line 43 of file navigation.h.

5.1.1.11 enum IDP::NavigationStatus

Current navigation status.

Enumerator:

NAVIGATION_ENROUTE
NAVIGATION_ARRIVED
NAVIGATION_LOST
MAX_STATUS

Definition at line 21 of file navigation.h.

5.1.1.12 enum IDP::NavigationTurn

Possible turns at a node.

Enumerator:

STRAIGHT
LEFT
RIGHT
BOTH
LEFT_AND_STRAIGHT
RIGHT_AND_STRAIGHT
BOTH_AND_STRAIGHT
END_OF_LINE
MAX_TURNS

Definition at line 51 of file navigation.h.

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.

Definition at line 36 of file line_following.h.

5.1.3.2 const double IDP::INTEGRAL_GAIN = 4.0

Constant for integral control in line following.

Definition at line 20 of file line_following.h.

5.1.3.3 const unsigned int IDP::LOST_TIMEOUT = 50

The number of loop iterations before we count as lost.

Definition at line 30 of file line_following.h.

5.1.3.4 const short unsigned int IDP::MAX_CORRECTION = 127

Maximum differential correction value before it gets capped.

Definition at line 25 of file line_following.h.

5.1.3.5 const int IDP::MOTOR_MAX_SPEED = 127

Highest allowable motor speed in either direction.

Definition at line 21 of file hal.h.

5.1.3.6 const int IDP::MOTOR_RAMP_TIME = 16

How fast to ramp the motors towards the desired speed.

Lower is faster.

Definition at line 27 of file hal.h.

5.1.3.7 const NavigationNode IDP::NAVIGATION_LOCATION_LOOKUP[MAX_LOCATION][2]

Initial value:

```
{
    {NODE9, NODE6},
    {NODE7, NODE10},
    {NODE3, NODE2}
}
```

The lookup table of NavigationLocations to a pair of NavigationNodes indicating the start and end node (with implied direction).

Definition at line 54 of file navigation.cc.

5.1.3.8 const NavigationTurn IDP::NAVIGATION_NODE_TURNS[MAX_DIRECTION][MAX_NODE]

Initial value:


```
{
    {STRAIGHT, BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, BOTH,
      BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT,
      BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, RIGHT, END_OF_LINE},
    {END_OF_LINE, BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, RIGHT_AND_STRAIGHT,
      BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT,
      BOTH_AND_STRAIGHT, BOTH_AND_STRAIGHT, LEFT, STRAIGHT}
}
```

The turns at each node.

Indexed by NavigationDirection and then by NavigationNode

Definition at line 29 of file navigation.cc.

5.1.3.9 const NavigationNode IDP::NAVIGATION_ROUTE_MAP[MAX_DIRECTION][MAX_NODE]

Initial value:

```
{
    {NODE2, NODE3, NODE4, NODE5, NODE6, NODE8, NODE8, NODE9, NODE10,
      NODE11, NODE10},
    {NODE2, NODE1, NODE2, NODE3, NODE4, NODE5, NODE6, NODE7, NODE8,
      NODE9, NODE10}
}
```

The route to take, node by node.

Definition at line 63 of file navigation.cc.

5.1.3.10 const NavigationTurn IDP::NAVIGATION_TURN_MAP[MAX_DIRECTION][MAX_NODE]

Initial value:

```
{
    {STRAIGHT, STRAIGHT, STRAIGHT, RIGHT, STRAIGHT, RIGHT, STRAIGHT,
      STRAIGHT, STRAIGHT, RIGHT, END_OF_LINE},
    {END_OF_LINE, STRAIGHT, STRAIGHT, LEFT, STRAIGHT, LEFT, STRAIGHT,
      STRAIGHT, STRAIGHT, LEFT, STRAIGHT}
}
```

Turns that should be taken at each node in each direction.

Indexed by NavigationDirection and then by NavigationNode

Definition at line 43 of file navigation.cc.

Chapter 6

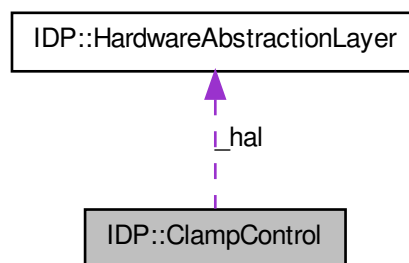
Class Documentation

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 (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.

- [BobbinColour colour](#) () const

Check the bobbin colour.

- [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.

Definition at line 39 of file clamp_control.h.

6.1.2 Constructor & Destructor Documentation

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

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

Parameters

hal A const pointer to an instance of the HAL

Definition at line 39 of file clamp_control.cc.

6.1.3 Member Function Documentation

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

Check the bobbin badness.

Returns

A BobbinBadness value to indicate current bobbin status

Definition at line 75 of file clamp_control.cc.

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

Check the bobbin colour.

Returns

A BobbinColour value to indicate current bobbin colour

Definition at line 65 of file clamp_control.cc.

6.1.3.3 void IDP::ClampControl::pick_up ()

Pick up something using the clamp.

Definition at line 48 of file clamp_control.cc.

6.1.3.4 void IDP::ClampControl::put_down ()

Put something in the clamp down.

Definition at line 56 of file clamp_control.cc.

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

- [src/libidp/clamp_control.h](#)
- [src/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)
Drive both motors forwards at a given speed.
- void [motors_backward](#) (const unsigned short int speed)
Drive both motors backwards at a given speed.
- void [motor_left_forward](#) (const unsigned short int speed)
Drive the left motor forward at the given speed.
- void [motor_right_forward](#) (const unsigned short int speed)
Drive the right motor forward at the given speed.
- void [motor_left_backward](#) (const unsigned short int speed)
Drive the left motor backward at the given speed.
- void [motor_right_backward](#) (const unsigned short int speed)
Drive the right motor backward at the given speed.
- void [motors_turn_left](#) (const unsigned short int speed)
Drive the motors to steer the robot to the left.
- void [motors_turn_right](#) (const unsigned short int speed)

Drive the motors to steer the robot to the right.

- void [motors_stop](#) ()
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)
Set the bobbin colour indication LEDs.
- void [colour_leds](#) (const bool red, const bool green)
Turn on and off the LEDs used to light up the bobbin for colour detection.
- void [bad_bobbin_led](#) (const bool status)
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)
Turn the grabber jaw actuator on or off.
- void [grabber_lift](#) (const bool status)
Turn the grabber lift mechanism actuator on or off.

6.2.1 Detailed Description

Provide a hardware agnostic interface to the required hardware functionality.

Definition at line 53 of file hal.h.

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.

Definition at line 31 of file hal.cc.

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

Definition at line 303 of file hal.cc.

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

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)

Definition at line 343 of file hal.cc.

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

Read the status register, discarding its value.

Definition at line 251 of file hal.cc.

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

Definition at line 293 of file hal.cc.

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

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)

Definition at line 330 of file hal.cc.

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

Turn the grabber jaw actuator on or off.

Parameters

status Jaw actuator status (true=on)

Definition at line 353 of file hal.cc.

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

Turn the grabber lift mechanism actuator on or off.

Parameters

status Lift actuator status (true=on)

Definition at line 363 of file hal.cc.

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

Definition at line 283 of file hal.cc.

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

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)

Definition at line 315 of file hal.cc.

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

Definition at line 214 of file hal.cc.

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

Drive the left motor backward at the given speed.

Parameters

speed The speed at which to drive the motor

Definition at line 132 of file hal.cc.

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

Drive the left motor forward at the given speed.

Parameters

speed The speed at which to drive the motor

Definition at line 98 of file hal.cc.

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

Drive the right motor backward at the given speed.

Parameters

speed The speed at which to drive the motor

Definition at line 149 of file hal.cc.

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

Drive the right motor forward at the given speed.

Parameters

speed The speed at which to drive the motor

Definition at line 115 of file hal.cc.

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

Drive both motors backwards at a given speed.

Parameters

speed The speed to drive at, 0 to 127

Definition at line 81 of file hal.cc.

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

Drive both motors forwards at a given speed.

Parameters

speed The speed to drive at, 0 to 127

Definition at line 64 of file hal.cc.

6.2.3.17 void IDP::HardwareAbstractionLayer::motors_stop ()

Stop all motors.

Definition at line 199 of file hal.cc.

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

Drive the motors to steer the robot to the left.

Parameters

speed The speed to drive at, 0 to 127

Definition at line 166 of file hal.cc.

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

Drive the motors to steer the robot to the right.

Parameters

speed The speed to drive at, 0 to 127

Definition at line 183 of file hal.cc.

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

Definition at line 273 of file hal.cc.

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

Read the status register and return it.

Returns

The STATUS register

Definition at line 262 of file hal.cc.

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

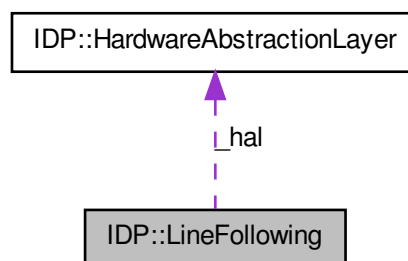
- src/libidp/[hal.h](#)
- src/libidp/[hal.cc](#)

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](#) ([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.
- [LineFollowingStatus turn_around_cw](#) (void)
Turn the robot around clockwise until the sensors detect another line.
- [LineFollowingStatus turn_around_ccw](#) (void)
Turn the robot around counterclockwise until the sensors detect another line.
- [LineFollowingStatus junction_status](#) (void)
Return whether we can see a junction or not, without changing motor settings.
- 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.

Definition at line 95 of file line_following.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 IDP::LineFollowing::LineFollowing ([HardwareAbstractionLayer](#) * *hal*)

Construct the Line Follower.

Definition at line 57 of file line_following.cc.

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.

Definition at line 71 of file line_following.cc.

6.3.3.2 LineFollowingStatus IDP::LineFollowing::junction_status (void)

Return whether we can see a junction or not, without changing motor settings.

Returns

A LineFollowingStatus indicating junctions or NO_TURNS_FOUND if no junctions found.

Definition at line 304 of file line_following.cc.

6.3.3.3 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.

Definition at line 339 of file line_following.cc.

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

Turn the robot around counterclockwise until the sensors detect another line.

Returns

A LineFollowingStatus code

Definition at line 292 of file line_following.cc.

6.3.3.5 LineFollowingStatus IDP::LineFollowing::turn_around_cw (void)

Turn the robot around clockwise until the sensors detect another line.

Returns

A LineFollowingStatus code

Definition at line 281 of file line_following.cc.

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

Turn the robot left until the sensors encounter another line.

Returns

A LineFollowingStatus code

Definition at line 261 of file line_following.cc.

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

Turn the robot right until the sensors detect another line.

Returns

A LineFollowingStatus code

Definition at line 271 of file line_following.cc.

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

- [src/libidp/line_following.h](#)
- [src/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.

Definition at line 41 of file hal.h.

6.4.2 Member Data Documentation

6.4.2.1 LineSensorStatus IDP::LineSensors::line_left

Definition at line 44 of file hal.h.

6.4.2.2 LineSensorStatus IDP::LineSensors::line_right

Definition at line 45 of file hal.h.

6.4.2.3 LineSensorStatus IDP::LineSensors::outer_left

Definition at line 43 of file hal.h.

6.4.2.4 LineSensorStatus IDP::LineSensors::outer_right

Definition at line 46 of file hal.h.

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

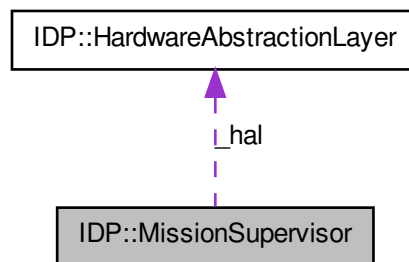
- src/libidp/[hal.h](#)

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](#).
- [~MissionSupervisor](#) ()
Destruct the [MissionSupervisor](#), deleting the HAL.
- 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.
- void [test_navigation](#) ()
Test navigation code.
- const [HardwareAbstractionLayer](#) * [hal](#) () const
Const accessor for the HAL.

6.5.1 Detailed Description

Control the overall robot behaviour and objective fulfillment.

Definition at line 24 of file mission_supervisor.h.

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

Definition at line 30 of file mission_supervisor.cc.

6.5.2.2 IDP::MissionSupervisor::~~MissionSupervisor ()

Destruct the [MissionSupervisor](#), deleting the HAL.

Definition at line 42 of file mission_supervisor.cc.

6.5.3 Member Function Documentation

6.5.3.1 void IDP::MissionSupervisor::drive_backward ()

Set both motors driving backwards.

Definition at line 70 of file mission_supervisor.cc.

6.5.3.2 void IDP::MissionSupervisor::drive_forward ()

Set both motors driving forwards.

Definition at line 59 of file mission_supervisor.cc.

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

Const accessor for the HAL.

Definition at line 256 of file mission_supervisor.cc.

6.5.3.4 void IDP::MissionSupervisor::run_task ()

Commence running the main task.

Definition at line 51 of file mission_supervisor.cc.

6.5.3.5 void IDP::MissionSupervisor::stop ()

Stop all motors.

Definition at line 80 of file mission_supervisor.cc.

6.5.3.6 void IDP::MissionSupervisor::test_line_following ()

Test line following on a straight line.

Definition at line 125 of file mission_supervisor.cc.

6.5.3.7 void IDP::MissionSupervisor::test_line_sensor ()

Attempt to read the line sensor status.

Definition at line 90 of file mission_supervisor.cc.

6.5.3.8 void IDP::MissionSupervisor::test_navigation ()

Test navigation code.

Definition at line 234 of file mission_supervisor.cc.

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

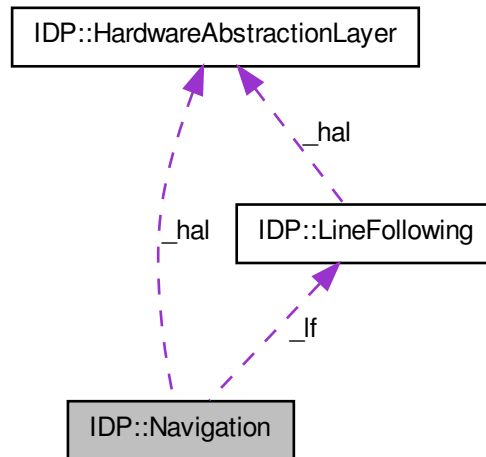
- src/libidp/[mission_supervisor.h](#)
- src/libidp/[mission_supervisor.cc](#)

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** ([HardwareAbstractionLayer](#) *hal, const [NavigationNode](#) from, const [NavigationNode](#) to)

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

- **~Navigation** ()

Destruct [Navigation](#), deleting the [LineFollowing](#) object.

- **NavigationStatus go** (const [NavigationLocation](#) location)

Go to a [NavigationLocation](#).

- **NavigationStatus go_node** (const [NavigationNode](#) target)

Go to a particular [NavigationNode](#).

6.6.1 Detailed Description

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

Definition at line 68 of file navigation.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 IDP::Navigation::Navigation (HardwareAbstractionLayer * *hal*, const NavigationNode *from* = *NODE7*, const NavigationNode *to* = *NODE8*)

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

The optional parameters *from* and *to* can be used to define the starting position, but default to the 'start box'.

Parameters

hal A const pointer to an instance of the HAL

from The node behind the robot at the start

to The node in front of the robot at the start

Definition at line 126 of file navigation.cc.

6.6.2.2 IDP::Navigation::~~Navigation ()

Destruct [Navigation](#), deleting the [LineFollowing](#) object.

Definition at line 143 of file navigation.cc.

6.6.3 Member Function Documentation

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

Go to a NavigationLocation.

Returns

A NavigationStatus code

Definition at line 155 of file navigation.cc.

6.6.3.2 NavigationStatus IDP::Navigation::go_node (const NavigationNode *target*)

Go to a particular NavigationNode.

Returns

A NavigationStatus code

Definition at line 166 of file navigation.cc.

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

- src/libidp/[navigation.h](#)
- src/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)
Construct 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.

Definition at line 21 of file `self_tests.h`.

6.7.2 Constructor & Destructor Documentation

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

Construct a SelfTest instance Completely separate 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

Definition at line 28 of file `self_tests.cc`.

6.7.3 Member Function Documentation

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

Fire each of the actuators in turn.

Definition at line 122 of file `self_tests.cc`.

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

Turn on the LED used for detecting bad bobbins.

Definition at line 192 of file `self_tests.cc`.

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

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

Definition at line 148 of file self_tests.cc.

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

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

Definition at line 139 of file self_tests.cc.

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

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

Definition at line 184 of file self_tests.cc.

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

Drive the robot backwards for a moment.

Definition at line 45 of file self_tests.cc.

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

Drive the robot forwards for a moment.

Definition at line 37 of file self_tests.cc.

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

Display the current ADC read from the light dependent resistor.

Definition at line 114 of file self_tests.cc.

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

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

Definition at line 130 of file self_tests.cc.

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.

Definition at line 98 of file self_tests.cc.

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

Display the state of each of the two microswitches.

Definition at line 106 of file self_tests.cc.

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.
Definition at line 157 of file self_tests.cc.

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.
Definition at line 166 of file self_tests.cc.

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

Turn on each of the status LEDs (used for indicating bobbin colour) in turn.
Definition at line 175 of file self_tests.cc.

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.
Definition at line 80 of file self_tests.cc.

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.
Definition at line 89 of file self_tests.cc.

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

Stop all of the robot's motors.
Definition at line 53 of file self_tests.cc.

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

Drive motors in opposite directions to turn the robot left on the spot.
Definition at line 62 of file self_tests.cc.

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

Drive motors in opposite directions to turn the robot right on the spot.
Definition at line 71 of file self_tests.cc.

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

- [src/libidp/self_tests.h](#)
- [src/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

- `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.

Definition at line 20 of file `status_watchdog.h`.

6.8.2 Member Function Documentation

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

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

Returns

The error encountered, if any

Definition at line 23 of file `status_watchdog.cc`.

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

- `src/libidp/status_watchdog.h`
- `src/libidp/status_watchdog.cc`

Chapter 7

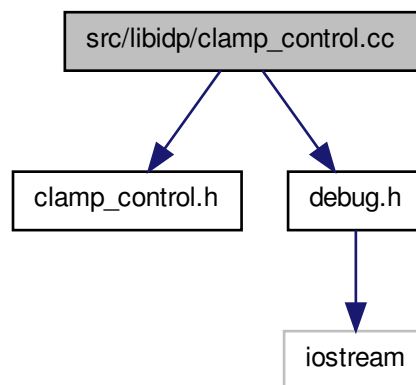
File Documentation

7.1 src/libidp/clamp_control.cc File Reference

```
#include "clamp_control.h"
```

```
#include "debug.h"
```

Include dependency graph for clamp_control.cc:



Namespaces

- namespace [IDP](#)

Defines

- `#define` [MODULE_NAME](#) "Clamp"

- `#define TRACE_ENABLED false`
- `#define DEBUG_ENABLED true`
- `#define INFO_ENABLED true`
- `#define ERROR_ENABLED true`

7.1.1 Define Documentation

7.1.1.1 `#define DEBUG_ENABLED true`

Definition at line 12 of file clamp_control.cc.

7.1.1.2 `#define ERROR_ENABLED true`

Definition at line 14 of file clamp_control.cc.

7.1.1.3 `#define INFO_ENABLED true`

Definition at line 13 of file clamp_control.cc.

7.1.1.4 `#define MODULE_NAME "Clamp"`

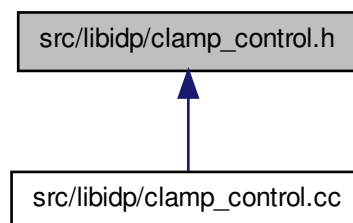
Definition at line 10 of file clamp_control.cc.

7.1.1.5 `#define TRACE_ENABLED false`

Definition at line 11 of file clamp_control.cc.

7.2 src/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.

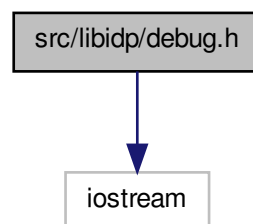
- enum [IDP::BobbinBadness](#) { [IDP::BOBBIN_GOOD](#), [IDP::BOBBIN_BAD](#) }

Bobbin good or bad.

7.3 src/libidp/debug.h File Reference

```
#include <iostream>
```

Include dependency graph for debug.h:



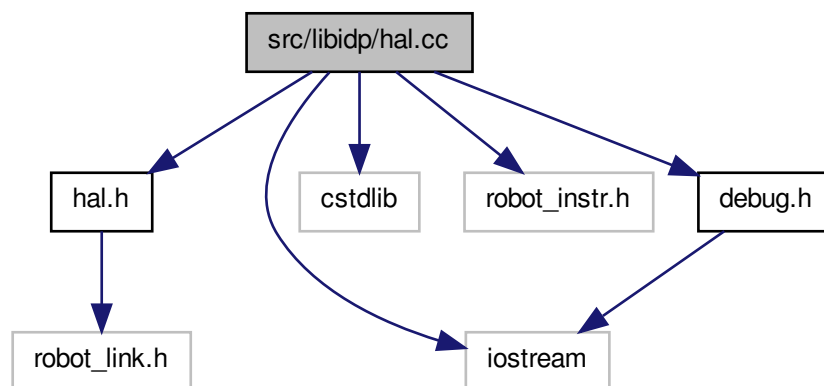
This graph shows which files directly or indirectly include this file:



7.4 src/libidp/hal.cc File Reference

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

Include dependency graph for hal.cc:



Namespaces

- namespace [IDP](#)

Defines

- `#define` [MODULE_NAME](#) "HAL"
- `#define` [TRACE_ENABLED](#) false
- `#define` [DEBUG_ENABLED](#) false
- `#define` [INFO_ENABLED](#) true
- `#define` [ERROR_ENABLED](#) true
- `#define` [UNUSED](#)(x) (void)(x)

7.4.1 Define Documentation

7.4.1.1 `#define` [DEBUG_ENABLED](#) false

Definition at line 17 of file `hal.cc`.

7.4.1.2 `#define ERROR_ENABLED true`

Definition at line 19 of file hal.cc.

7.4.1.3 `#define INFO_ENABLED true`

Definition at line 18 of file hal.cc.

7.4.1.4 `#define MODULE_NAME "HAL"`

Definition at line 15 of file hal.cc.

7.4.1.5 `#define TRACE_ENABLED false`

Definition at line 16 of file hal.cc.

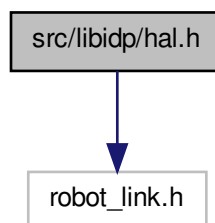
7.4.1.6 `#define UNUSED(x) (void)(x)`

Definition at line 23 of file hal.cc.

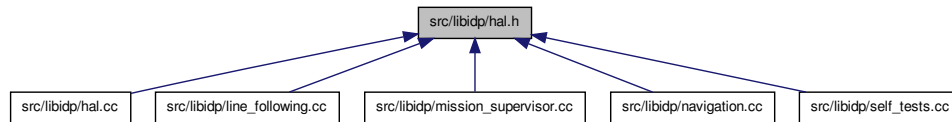
7.5 src/libidp/hal.h File Reference

```
#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`.

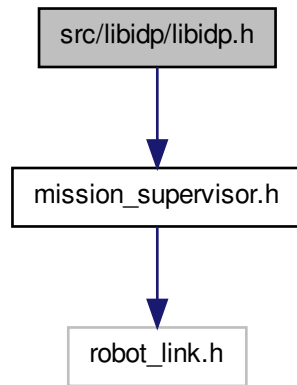
Variables

- 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.

7.6 src/libidp/libidp.h File Reference

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

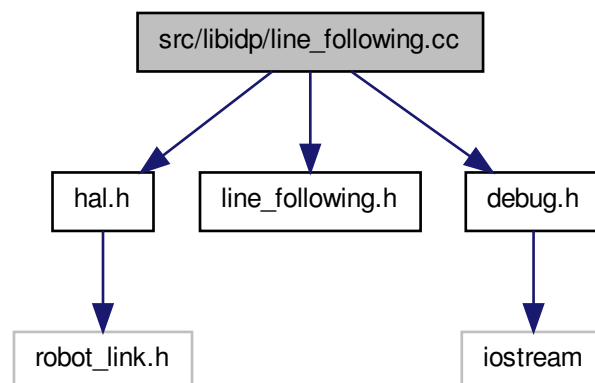
Include dependency graph for libidp.h:



7.7 src/libidp/line_following.cc File Reference

```
#include "hal.h"  
#include "line_following.h"  
#include "debug.h"
```

Include dependency graph for line_following.cc:



Namespaces

- namespace [IDP](#)

Defines

- #define [MODULE_NAME](#) "LineFollowing"
- #define [TRACE_ENABLED](#) false
- #define [DEBUG_ENABLED](#) true
- #define [INFO_ENABLED](#) true
- #define [ERROR_ENABLED](#) true

Functions

- unsigned short int [IDP::cap_correction](#) (const unsigned short int correction)
Cap a line following correction value to MAX_CORRECTION.

7.7.1 Define Documentation

7.7.1.1 #define [DEBUG_ENABLED](#) true

Definition at line 13 of file line_following.cc.

7.7.1.2 #define [ERROR_ENABLED](#) true

Definition at line 15 of file line_following.cc.

7.7.1.3 #define [INFO_ENABLED](#) true

Definition at line 14 of file line_following.cc.

7.7.1.4 #define [MODULE_NAME](#) "LineFollowing"

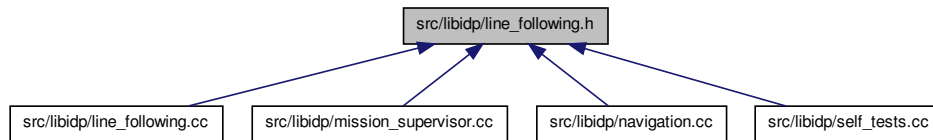
Definition at line 11 of file line_following.cc.

7.7.1.5 #define [TRACE_ENABLED](#) false

Definition at line 12 of file line_following.cc.

7.8 src/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 manoeuvring.

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](#), [IDP::NO_TURNS_FOUND](#) }
Line following return status codes.
- enum [IDP::LineFollowingTurnDirection](#) {
[IDP::TURN_LEFT](#), [IDP::TURN_RIGHT](#), [IDP::TURN_AROUND_CW](#), [IDP::TURN_AROUND_CCW](#),
[IDP::MAX_TURN_DIRECTION](#) }
Possible turn directions.
- enum [IDP::LineFollowingLineStatus](#) { [IDP::ON_LINE](#), [IDP::LOST_LINE](#), [IDP::OTHER](#),
[IDP::MAX_LINE_STATUS](#) }
Possible line statuses, used internally.

Functions

- unsigned short int [IDP::cap_correction](#) (const unsigned short int correction)
Cap a line following correction value to MAX_CORRECTION.

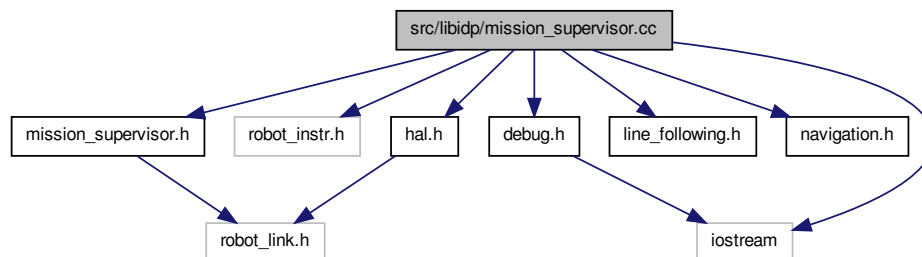
Variables

- 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.

7.9 src/libidp/mission_supervisor.cc File Reference

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

Include dependency graph for mission_supervisor.cc:



Namespaces

- namespace `IDP`

Defines

- `#define MODULE_NAME "MisSup"`

- #define `TRACE_ENABLED` false
- #define `DEBUG_ENABLED` true
- #define `INFO_ENABLED` true
- #define `ERROR_ENABLED` true

7.9.1 Define Documentation

7.9.1.1 #define `DEBUG_ENABLED` true

Definition at line 18 of file mission_supervisor.cc.

7.9.1.2 #define `ERROR_ENABLED` true

Definition at line 20 of file mission_supervisor.cc.

7.9.1.3 #define `INFO_ENABLED` true

Definition at line 19 of file mission_supervisor.cc.

7.9.1.4 #define `MODULE_NAME` "MisSup"

Definition at line 16 of file mission_supervisor.cc.

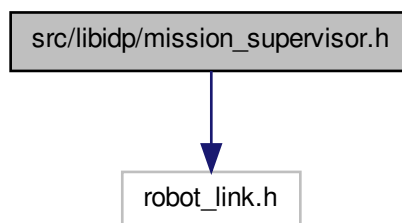
7.9.1.5 #define `TRACE_ENABLED` false

Definition at line 17 of file mission_supervisor.cc.

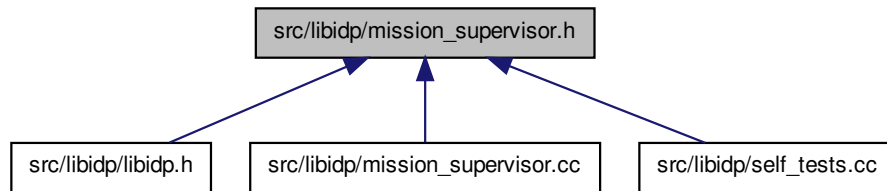
7.10 src/libidp/mission_supervisor.h File Reference

```
#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.

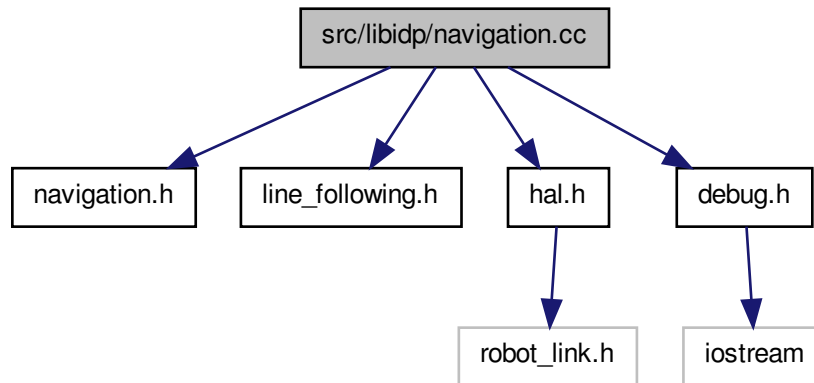
Namespaces

- namespace [IDP](#)

7.11 src/libdp/navigation.cc File Reference

```
#include "navigation.h"
#include "line_following.h"
#include "hal.h"
#include "debug.h"
```

Include dependency graph for navigation.cc:



Namespaces

- namespace `IDP`

Defines

- `#define` `MODULE_NAME` "Navigation"
- `#define` `TRACE_ENABLED` false
- `#define` `DEBUG_ENABLED` true
- `#define` `INFO_ENABLED` true
- `#define` `ERROR_ENABLED` true
- `#define` `UNUSED(x)` (void)(x)

Variables

- `const` `NavigationTurn` `IDP::NAVIGATION_NODE_TURNS` `[MAX_DIRECTION][MAX_NODE]`
The turns at each node.
- `const` `NavigationTurn` `IDP::NAVIGATION_TURN_MAP` `[MAX_DIRECTION][MAX_NODE]`
Turns that should be taken at each node in each direction.
- `const` `NavigationNode` `IDP::NAVIGATION_LOCATION_LOOKUP` `[MAX_LOCATION][2]`
The lookup table of NavigationLocations to a pair of NavigationNodes indicating the start and end node (with implied direction).
- `const` `NavigationNode` `IDP::NAVIGATION_ROUTE_MAP` `[MAX_DIRECTION][MAX_NODE]`
The route to take, node by node.

7.11.1 Define Documentation

7.11.1.1 `#define DEBUG_ENABLED true`

Definition at line 14 of file navigation.cc.

7.11.1.2 `#define ERROR_ENABLED true`

Definition at line 16 of file navigation.cc.

7.11.1.3 `#define INFO_ENABLED true`

Definition at line 15 of file navigation.cc.

7.11.1.4 `#define MODULE_NAME "Navigation"`

Definition at line 12 of file navigation.cc.

7.11.1.5 `#define TRACE_ENABLED false`

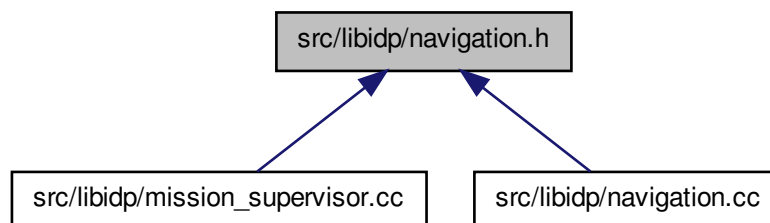
Definition at line 13 of file navigation.cc.

7.11.1.6 `#define UNUSED(x) (void)(x)`

Definition at line 20 of file navigation.cc.

7.12 src/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](#), [IDP::MAX_STATUS](#) }

Current navigation status.

- enum [IDP::NavigationLocation](#) { [IDP::NAVIGATION_BOXES](#), [IDP::NAVIGATION_RACK](#), [IDP::NAVIGATION_DELIVERY](#), [IDP::MAX_LOCATION](#) }

Possible locations for navigation to be asked to go to.

- enum [IDP::NavigationDirection](#) { [IDP::NAVIGATION_CLOCKWISE](#), [IDP::NAVIGATION_ANTI_CLOCKWISE](#), [IDP::MAX_DIRECTION](#) }

Directions around the circuit.

- enum [IDP::NavigationNode](#) { [IDP::NODE1](#), [IDP::NODE2](#), [IDP::NODE3](#), [IDP::NODE4](#), [IDP::NODE5](#), [IDP::NODE6](#), [IDP::NODE7](#), [IDP::NODE8](#), [IDP::NODE9](#), [IDP::NODE10](#), [IDP::NODE11](#), [IDP::MAX_NODE](#) }

Navigation nodes, numbered clockwise from the bottom right corner of the table.

- enum [IDP::NavigationTurn](#) { [IDP::STRAIGHT](#), [IDP::LEFT](#), [IDP::RIGHT](#), [IDP::BOTH](#), [IDP::LEFT_AND_STRAIGHT](#), [IDP::RIGHT_AND_STRAIGHT](#), [IDP::BOTH_AND_STRAIGHT](#), [IDP::END_OF_LINE](#), [IDP::MAX_TURNS](#) }

Possible turns at a node.

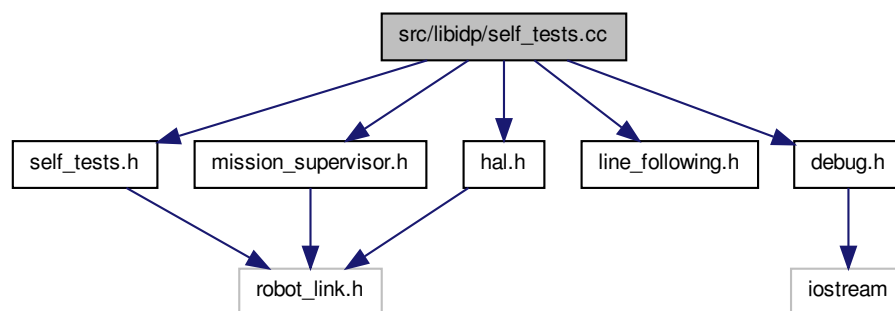
- enum [IDP::NavigationCachedJunction](#) { [IDP::NO_CACHE](#), [IDP::LEFT_TURN](#), [IDP::RIGHT_TURN](#), [IDP::BOTH_TURNS](#), [IDP::NO_TURNS](#) }

Cached junction information for use while driving over a junction or executing a turn.

7.13 src/libidp/self_tests.cc File Reference

```
#include "self_tests.h"
#include "mission_supervisor.h"
#include "hal.h"
#include "line_following.h"
#include "debug.h"
```

Include dependency graph for self_tests.cc:



Namespaces

- namespace [IDP](#)

Defines

- `#define` [MODULE_NAME](#) "SelfTests"
- `#define` [TRACE_ENABLED](#) false
- `#define` [DEBUG_ENABLED](#) true
- `#define` [INFO_ENABLED](#) true
- `#define` [ERROR_ENABLED](#) true

7.13.1 Define Documentation

7.13.1.1 `#define` [DEBUG_ENABLED](#) true

Definition at line 16 of file `self_tests.cc`.

7.13.1.2 `#define` [ERROR_ENABLED](#) true

Definition at line 18 of file `self_tests.cc`.

7.13.1.3 `#define INFO_ENABLED true`

Definition at line 17 of file self_tests.cc.

7.13.1.4 `#define MODULE_NAME "SelfTests"`

Definition at line 14 of file self_tests.cc.

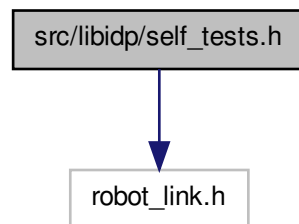
7.13.1.5 `#define TRACE_ENABLED false`

Definition at line 15 of file self_tests.cc.

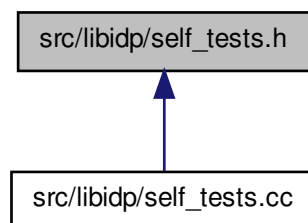
7.14 src/libidp/self_tests.h File Reference

```
#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

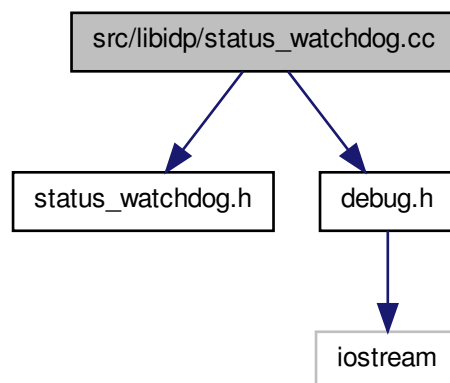
- namespace [IDP](#)

7.15 src/libidp/status_watchdog.cc File Reference

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

```
#include "debug.h"
```

Include dependency graph for status_watchdog.cc:



Namespaces

- namespace [IDP](#)

Defines

- `#define` [MODULE_NAME](#) "StatusWatch"
- `#define` [TRACE_ENABLED](#) false
- `#define` [DEBUG_ENABLED](#) true
- `#define` [INFO_ENABLED](#) true
- `#define` [ERROR_ENABLED](#) true

7.15.1 Define Documentation

7.15.1.1 `#define DEBUG_ENABLED true`

Definition at line 13 of file status_watchdog.cc.

7.15.1.2 `#define ERROR_ENABLED true`

Definition at line 15 of file status_watchdog.cc.

7.15.1.3 `#define INFO_ENABLED true`

Definition at line 14 of file status_watchdog.cc.

7.15.1.4 `#define MODULE_NAME "StatusWatch"`

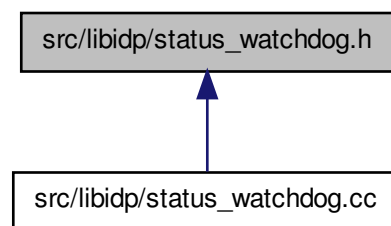
Definition at line 11 of file status_watchdog.cc.

7.15.1.5 `#define TRACE_ENABLED false`

Definition at line 12 of file status_watchdog.cc.

7.16 src/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

- ~MissionSupervisor
 - IDP::MissionSupervisor, [32](#)
- ~Navigation
 - IDP::Navigation, [35](#)
- ACTION_COMPLETED
 - IDP, [12](#)
- ACTION_IN_PROGRESS
 - IDP, [12](#)
- actuators
 - IDP::SelfTests, [37](#)
- bad_bobbin_ldr
 - IDP::HardwareAbstractionLayer, [23](#)
- bad_bobbin_led
 - IDP::HardwareAbstractionLayer, [23](#)
- badness
 - IDP::ClampControl, [20](#)
- badness_LED
 - IDP::SelfTests, [37](#)
- BOBBIN_BAD
 - IDP, [12](#)
- BOBBIN_GOOD
 - IDP, [12](#)
- BOBBIN_GREEN
 - IDP, [12](#)
- BOBBIN_RED
 - IDP, [12](#)
- BOBBIN_WHITE
 - IDP, [12](#)
- bobbin_analyse
 - IDP::SelfTests, [37](#)
- BobbinBadness
 - IDP, [12](#)
- BobbinColour
 - IDP, [12](#)
- BOTH
 - IDP, [15](#)
- BOTH_AND_STRAIGHT
 - IDP, [15](#)
- BOTH_TURNS
 - IDP, [13](#)
- BOTH_TURNS_FOUND
 - IDP, [13](#)
- cap_correction
 - IDP, [15](#)
- check
 - IDP::StatusWatchdog, [40](#)
- clamp_control
 - IDP::SelfTests, [38](#)
- clamp_control.cc
 - DEBUG_ENABLED, [42](#)
 - ERROR_ENABLED, [42](#)
 - INFO_ENABLED, [42](#)
 - MODULE_NAME, [42](#)
 - TRACE_ENABLED, [42](#)
- ClampControl
 - IDP::ClampControl, [20](#)
- clear_status_register
 - IDP::HardwareAbstractionLayer, [23](#)
- colour
 - IDP::ClampControl, [20](#)
- colour_ldr
 - IDP::HardwareAbstractionLayer, [23](#)
- colour_leds
 - IDP::HardwareAbstractionLayer, [23](#)
- colour_sensor_LEDs
 - IDP::SelfTests, [38](#)
- DEBUG_ENABLED
 - clamp_control.cc, [42](#)
 - hal.cc, [44](#)
 - line_following.cc, [48](#)
 - mission_supervisor.cc, [51](#)
 - navigation.cc, [54](#)
 - self_tests.cc, [56](#)
 - status_watchdog.cc, [59](#)
- drive_backward
 - IDP::MissionSupervisor, [32](#)
 - IDP::SelfTests, [38](#)
- drive_forward
 - IDP::MissionSupervisor, [32](#)
 - IDP::SelfTests, [38](#)
- EDGE_ERROR
 - IDP, [15](#)
- END_OF_LINE
 - IDP, [15](#)
- ERROR_ENABLED
 - clamp_control.cc, [42](#)

- hal.cc, 44
- line_following.cc, 48
- mission_supervisor.cc, 51
- navigation.cc, 54
- self_tests.cc, 56
- status_watchdog.cc, 59
- follow_line
 - IDP::LineFollowing, 28
- go
 - IDP::Navigation, 35
- go_node
 - IDP::Navigation, 35
- grabber_jaw
 - IDP::HardwareAbstractionLayer, 24
- grabber_lift
 - IDP::HardwareAbstractionLayer, 24
- grabber_switch
 - IDP::HardwareAbstractionLayer, 24
- hal
 - IDP::MissionSupervisor, 32
- hal.cc
 - DEBUG_ENABLED, 44
 - ERROR_ENABLED, 44
 - INFO_ENABLED, 45
 - MODULE_NAME, 45
 - TRACE_ENABLED, 45
 - UNUSED, 45
- HardwareAbstractionLayer
 - IDP::HardwareAbstractionLayer, 23
- IDP, 9
 - ACTION_COMPLETED, 12
 - ACTION_IN_PROGRESS, 12
 - BOBBIN_BAD, 12
 - BOBBIN_GOOD, 12
 - BOBBIN_GREEN, 12
 - BOBBIN_RED, 12
 - BOBBIN_WHITE, 12
 - BobbinBadness, 12
 - BobbinColour, 12
 - BOTH, 15
 - BOTH_AND_STRAIGHT, 15
 - BOTH_TURNS, 13
 - BOTH_TURNS_FOUND, 13
 - cap_correction, 15
 - EDGE_ERROR, 15
 - END_OF_LINE, 15
 - INTEGRAL_GAIN, 15
 - LEFT, 15
 - LEFT_AND_STRAIGHT, 15
 - LEFT_TURN, 13
 - LEFT_TURN_FOUND, 13
 - LINE, 13
 - LineFollowingLineStatus, 12
 - LineFollowingStatus, 12
 - LineFollowingTurnDirection, 13
 - LineSensorStatus, 13
 - LOST, 13
 - LOST_LINE, 12
 - LOST_TIMEOUT, 16
 - MAX_DIRECTION, 14
 - MAX_LINE_STATUS, 12
 - MAX_LOCATION, 14
 - MAX_NODE, 14
 - MAX_STATUS, 15
 - MAX_TURN_DIRECTION, 13
 - MAX_TURNS, 15
 - MAX_CORRECTION, 16
 - MOTOR_MAX_SPEED, 16
 - MOTOR_RAMP_TIME, 16
 - NAVIGATION_ANTICLOCKWISE, 14
 - NAVIGATION_ARRIVED, 15
 - NAVIGATION_BOXES, 14
 - NAVIGATION_CLOCKWISE, 14
 - NAVIGATION_DELIVERY, 14
 - NAVIGATION_ENROUTE, 15
 - NAVIGATION_LOST, 15
 - NAVIGATION_RACK, 14
 - NAVIGATION_LOCATION_LOOKUP, 16
 - NAVIGATION_NODE_TURNS, 16
 - NAVIGATION_ROUTE_MAP, 17
 - NAVIGATION_TURN_MAP, 17
 - NavigationCachedJunction, 13
 - NavigationDirection, 13
 - NavigationLocation, 14
 - NavigationNode, 14
 - NavigationStatus, 14
 - NavigationTurn, 15
 - NO_CACHE, 13
 - NO_LINE, 13
 - NO_TURNS, 13
 - NO_TURNS_FOUND, 13
 - NODE1, 14
 - NODE10, 14
 - NODE11, 14
 - NODE2, 14
 - NODE3, 14
 - NODE4, 14
 - NODE5, 14
 - NODE6, 14
 - NODE7, 14
 - NODE8, 14
 - NODE9, 14
 - ON_LINE, 12
 - OTHER, 12

- RIGHT, 15
- RIGHT_AND_STRAIGHT, 15
- RIGHT_TURN, 13
- RIGHT_TURN_FOUND, 13
- STRAIGHT, 15
- TURN_AROUND_CCW, 13
- TURN_AROUND_CW, 13
- TURN_LEFT, 13
- TURN_RIGHT, 13
- IDP::ClampControl, 19
 - badness, 20
 - ClampControl, 20
 - colour, 20
 - pick_up, 20
 - put_down, 21
- IDP::HardwareAbstractionLayer, 21
 - bad_bobbin_ldr, 23
 - bad_bobbin_led, 23
 - clear_status_register, 23
 - colour_ldr, 23
 - colour_leds, 23
 - grabber_jaw, 24
 - grabber_lift, 24
 - grabber_switch, 24
 - HardwareAbstractionLayer, 23
 - indication_LEDs, 24
 - line_following_sensors, 24
 - motor_left_backward, 25
 - motor_left_forward, 25
 - motor_right_backward, 25
 - motor_right_forward, 25
 - motors_backward, 25
 - motors_forward, 26
 - motors_stop, 26
 - motors_turn_left, 26
 - motors_turn_right, 26
 - reset_switch, 26
 - status_register, 27
- IDP::LineFollowing, 27
 - follow_line, 28
 - junction_status, 29
 - LineFollowing, 28
 - set_speed, 29
 - turn_around_ccw, 29
 - turn_around_cw, 29
 - turn_left, 29
 - turn_right, 29
- IDP::LineSensors, 30
 - line_left, 30
 - line_right, 30
 - outer_left, 30
 - outer_right, 30
- IDP::MissionSupervisor, 31
 - ~MissionSupervisor, 32
 - drive_backward, 32
 - drive_forward, 32
 - hal, 32
 - MissionSupervisor, 32
 - run_task, 33
 - stop, 33
 - test_line_following, 33
 - test_line_sensor, 33
 - test_navigation, 33
- IDP::Navigation, 33
 - ~Navigation, 35
 - go, 35
 - go_node, 35
 - Navigation, 35
- IDP::SelfTests, 36
 - actuators, 37
 - badness_LED, 37
 - bobbin_analyse, 37
 - clamp_control, 38
 - colour_sensor_LEDs, 38
 - drive_backward, 38
 - drive_forward, 38
 - LDRs, 38
 - line_following, 38
 - line_sensors, 38
 - microswitches, 38
 - navigate, 38
 - position, 39
 - SelfTests, 37
 - status_LEDs, 39
 - steer_left, 39
 - steer_right, 39
 - stop, 39
 - turn_left, 39
 - turn_right, 39
- IDP::StatusWatchdog, 40
 - check, 40
- indication_LEDs
 - IDP::HardwareAbstractionLayer, 24
- INFO_ENABLED
 - clamp_control.cc, 42
 - hal.cc, 45
 - line_following.cc, 48
 - mission_supervisor.cc, 51
 - navigation.cc, 54
 - self_tests.cc, 56
 - status_watchdog.cc, 59
- INTEGRAL_GAIN
 - IDP, 15
- junction_status
 - IDP::LineFollowing, 29
- LDRs

- IDP::SelfTests, 38
- LEFT
 - IDP, 15
- LEFT_AND_STRAIGHT
 - IDP, 15
- LEFT_TURN
 - IDP, 13
- LEFT_TURN_FOUND
 - IDP, 13
- LINE
 - IDP, 13
- line_following
 - IDP::SelfTests, 38
- line_following.cc
 - DEBUG_ENABLED, 48
 - ERROR_ENABLED, 48
 - INFO_ENABLED, 48
 - MODULE_NAME, 48
 - TRACE_ENABLED, 48
- line_following_sensors
 - IDP::HardwareAbstractionLayer, 24
- line_left
 - IDP::LineSensors, 30
- line_right
 - IDP::LineSensors, 30
- line_sensors
 - IDP::SelfTests, 38
- LineFollowing
 - IDP::LineFollowing, 28
- LineFollowingLineStatus
 - IDP, 12
- LineFollowingStatus
 - IDP, 12
- LineFollowingTurnDirection
 - IDP, 13
- LineSensorStatus
 - IDP, 13
- LOST
 - IDP, 13
- LOST_LINE
 - IDP, 12
- LOST_TIMEOUT
 - IDP, 16
- MAX_DIRECTION
 - IDP, 14
- MAX_LINE_STATUS
 - IDP, 12
- MAX_LOCATION
 - IDP, 14
- MAX_NODE
 - IDP, 14
- MAX_STATUS
 - IDP, 15
- MAX_TURN_DIRECTION
 - IDP, 13
- MAX_TURNS
 - IDP, 15
- MAX_CORRECTION
 - IDP, 16
- microswitches
 - IDP::SelfTests, 38
- mission_supervisor.cc
 - DEBUG_ENABLED, 51
 - ERROR_ENABLED, 51
 - INFO_ENABLED, 51
 - MODULE_NAME, 51
 - TRACE_ENABLED, 51
- MissionSupervisor
 - IDP::MissionSupervisor, 32
- MODULE_NAME
 - clamp_control.cc, 42
 - hal.cc, 45
 - line_following.cc, 48
 - mission_supervisor.cc, 51
 - navigation.cc, 54
 - self_tests.cc, 57
 - status_watchdog.cc, 59
- motor_left_backward
 - IDP::HardwareAbstractionLayer, 25
- motor_left_forward
 - IDP::HardwareAbstractionLayer, 25
- MOTOR_MAX_SPEED
 - IDP, 16
- MOTOR_RAMP_TIME
 - IDP, 16
- motor_right_backward
 - IDP::HardwareAbstractionLayer, 25
- motor_right_forward
 - IDP::HardwareAbstractionLayer, 25
- motors_backward
 - IDP::HardwareAbstractionLayer, 25
- motors_forward
 - IDP::HardwareAbstractionLayer, 26
- motors_stop
 - IDP::HardwareAbstractionLayer, 26
- motors_turn_left
 - IDP::HardwareAbstractionLayer, 26
- motors_turn_right
 - IDP::HardwareAbstractionLayer, 26
- navigate
 - IDP::SelfTests, 38
- Navigation
 - IDP::Navigation, 35
- navigation.cc
 - DEBUG_ENABLED, 54
 - ERROR_ENABLED, 54

- INFO_ENABLED, [54](#)
- MODULE_NAME, [54](#)
- TRACE_ENABLED, [54](#)
- UNUSED, [54](#)
- NAVIGATION_ANTICLOCKWISE
 - IDP, [14](#)
- NAVIGATION_ARRIVED
 - IDP, [15](#)
- NAVIGATION_BOXES
 - IDP, [14](#)
- NAVIGATION_CLOCKWISE
 - IDP, [14](#)
- NAVIGATION_DELIVERY
 - IDP, [14](#)
- NAVIGATION_ENROUTE
 - IDP, [15](#)
- NAVIGATION_LOST
 - IDP, [15](#)
- NAVIGATION_RACK
 - IDP, [14](#)
- NAVIGATION_LOCATION_LOOKUP
 - IDP, [16](#)
- NAVIGATION_NODE_TURNS
 - IDP, [16](#)
- NAVIGATION_ROUTE_MAP
 - IDP, [17](#)
- NAVIGATION_TURN_MAP
 - IDP, [17](#)
- NavigationCachedJunction
 - IDP, [13](#)
- NavigationDirection
 - IDP, [13](#)
- NavigationLocation
 - IDP, [14](#)
- NavigationNode
 - IDP, [14](#)
- NavigationStatus
 - IDP, [14](#)
- NavigationTurn
 - IDP, [15](#)
- NO_CACHE
 - IDP, [13](#)
- NO_LINE
 - IDP, [13](#)
- NO_TURNS
 - IDP, [13](#)
- NO_TURNS_FOUND
 - IDP, [13](#)
- NODE1
 - IDP, [14](#)
- NODE10
 - IDP, [14](#)
- NODE11
 - IDP, [14](#)
- NODE2
 - IDP, [14](#)
- NODE3
 - IDP, [14](#)
- NODE4
 - IDP, [14](#)
- NODE5
 - IDP, [14](#)
- NODE6
 - IDP, [14](#)
- NODE7
 - IDP, [14](#)
- NODE8
 - IDP, [14](#)
- NODE9
 - IDP, [14](#)
- ON_LINE
 - IDP, [12](#)
- OTHER
 - IDP, [12](#)
- outer_left
 - IDP::LineSensors, [30](#)
- outer_right
 - IDP::LineSensors, [30](#)
- pick_up
 - IDP::ClampControl, [20](#)
- position
 - IDP::SelfTests, [39](#)
- put_down
 - IDP::ClampControl, [21](#)
- reset_switch
 - IDP::HardwareAbstractionLayer, [26](#)
- RIGHT
 - IDP, [15](#)
- RIGHT_AND_STRAIGHT
 - IDP, [15](#)
- RIGHT_TURN
 - IDP, [13](#)
- RIGHT_TURN_FOUND
 - IDP, [13](#)
- run_task
 - IDP::MissionSupervisor, [33](#)
- self_tests.cc
 - DEBUG_ENABLED, [56](#)
 - ERROR_ENABLED, [56](#)
 - INFO_ENABLED, [56](#)
 - MODULE_NAME, [57](#)
 - TRACE_ENABLED, [57](#)
- SelfTests
 - IDP::SelfTests, [37](#)

set_speed
 IDP::LineFollowing, 29
 src/libidp/clamp_control.cc, 41
 src/libidp/clamp_control.h, 42
 src/libidp/debug.h, 43
 src/libidp/hal.cc, 44
 src/libidp/hal.h, 45
 src/libidp/libidp.h, 46
 src/libidp/line_following.cc, 47
 src/libidp/line_following.h, 49
 src/libidp/mission_supervisor.cc, 50
 src/libidp/mission_supervisor.h, 51
 src/libidp/navigation.cc, 52
 src/libidp/navigation.h, 54
 src/libidp/self_tests.cc, 56
 src/libidp/self_tests.h, 57
 src/libidp/status_watchdog.cc, 58
 src/libidp/status_watchdog.h, 59
 status_LEDs
 IDP::SelfTests, 39
 status_register
 IDP::HardwareAbstractionLayer, 27
 status_watchdog.cc
 DEBUG_ENABLED, 59
 ERROR_ENABLED, 59
 INFO_ENABLED, 59
 MODULE_NAME, 59
 TRACE_ENABLED, 59
 steer_left
 IDP::SelfTests, 39
 steer_right
 IDP::SelfTests, 39
 stop
 IDP::MissionSupervisor, 33
 IDP::SelfTests, 39
 STRAIGHT
 IDP, 15

 test_line_following
 IDP::MissionSupervisor, 33
 test_line_sensor
 IDP::MissionSupervisor, 33
 test_navigation
 IDP::MissionSupervisor, 33
 TRACE_ENABLED
 clamp_control.cc, 42
 hal.cc, 45
 line_following.cc, 48
 mission_supervisor.cc, 51
 navigation.cc, 54
 self_tests.cc, 57
 status_watchdog.cc, 59
 TURN_AROUND_CCW
 IDP, 13

 TURN_AROUND_CW
 IDP, 13
 TURN_LEFT
 IDP, 13
 TURN_RIGHT
 IDP, 13
 turn_around_ccw
 IDP::LineFollowing, 29
 turn_around_cw
 IDP::LineFollowing, 29
 turn_left
 IDP::LineFollowing, 29
 IDP::SelfTests, 39
 turn_right
 IDP::LineFollowing, 29
 IDP::SelfTests, 39

 UNUSED
 hal.cc, 45
 navigation.cc, 54