MComp Mapping

Generated by Doxygen 1.9.4

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 line_equation Struct Reference

Data Fields

- long double **m**
- long double **c**

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

- Traversal/map.h

3.2 point Struct Reference

Data Fields

- long double **x**
- long double **y**

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

- Traversal/map.h

3.3 queue Struct Reference

Data Fields

- int8_t **inside** [QUEUE_LEN]
- int8_t **length**

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

- Traversal/queue.h

Chapter 4

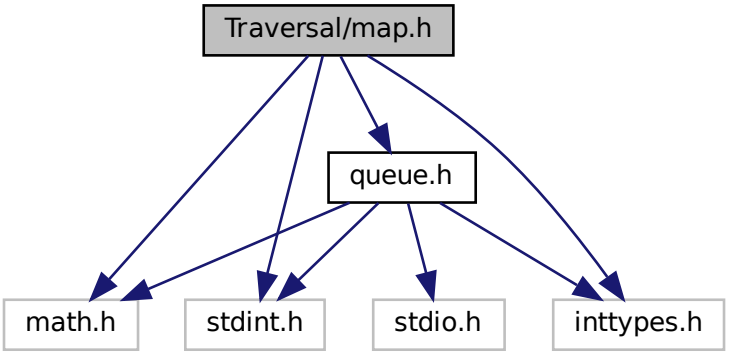
File Documentation

4.1 Traversal/map.h File Reference

The main functions for mapping the robot to the given route.

```
#include <math.h>
#include <stdint.h>
#include <inttypes.h>
#include "queue.h"
```

Include dependency graph for map.h:



Data Structures

- struct line_equation
- struct point

Macros

- #define **M_PI** 3.14159265358979323846
- #define **FLT_EPSILON** 1.19209290E-07F

Functions

- uint8_t get_line (struct line_equation *line, struct point *a, struct point *b)

  A helper function to return the line equation between two points.
- long double distance_from_line (struct line_equation *line, struct point *a)

  A helper function to determine a points distance from a given line.
- int8_t is_next_point (struct point *a, struct point *next)

  Determines the robot has reached the next checkpoint in route.
- int8_t is_outside_buffer (struct line_equation *line, struct point *a)

  A helper function to determine if the robot is outside the inaccuracy boundary.
- int8_t is_possible (struct point *a, struct point *b, float time)

  Determines if the movement between points is physically possible.

4.1.1 Detailed Description

The main functions for mapping the robot to the given route.

Author

T. Buckingham

Date

Thu Feb 16 11:00:05 2023

The functions used to determine the line between two points, the robot's distance from the given line, and other error checking functions such as impossible movement.

4.1.2 Function Documentation

4.1.2.1 distance_from_line()

```
long double distance_from_line (
            struct line_equation * line,
            struct point * a )
```

A helper function to determine a points distance from a given line.

Parameters

line	The line the robot is currently traversing.
a	The robot's current position.

Returns

The distance a is from line

4.1.2.2 get_line()

```
uint8_t get_line (
            struct line_equation * line,
            struct point * a,
            struct point * b )
```

A helper function to return the line equation between two points.

Parameters

line	The line struct to hold the values.
a	Point a on a plane
b	Pount b on a plane

Returns

Error checking value.

4.1.2.3 is_next_point()

```
int8_t is_next_point (
            struct point * a,
            struct point * next )
```

Determines the robot has reached the next checkpoint in route.

Parameters

a	The robot's current position.
next	The next point the robot is currently heading towards.

Returns

Wether the next point has been reached or not.

4.1.2.4 is_outside_buffer()

```
int8_t is_outside_buffer (
            struct line_equation * line,
            struct point * a )
```

A helper function to determine if the robot is outside the inaccuracy boundary.

Parameters

line	The line the robot is currently traversing.
a	The robot's current position.

Returns

Whether the robot is outside the given buffer or not.

4.1.2.5 is_possible()

```
int8_t is_possible (
            struct point * a,
            struct point * b,
            float time )
```

Determines if the movement between points is physically possible.

Given the speed, time and distance; is it possible for the robot to move in such a way.

Parameters

a	The robot's previous location.
b	The robot's current position.
time	The time between point a and b.

Returns

Whether the movement was possible or not.

4.2 map.h

Go to the documentation of this file.

```
1
15 #ifndef MAP_H
16 #define MAP_H
17
18 #include <math.h>
19 #include <stdint.h>
20 #include <inttypes.h>
21 #include "queue.h"
22
23
24 #ifndef M_PI
25 #define M_PI 3.14159265358979323846
26 #endif
27
28 #ifndef FLT_EPSILON
29 #define FLT_EPSILON 1.19209290E-07F
30 #endif
31
32 struct line_equation {
```

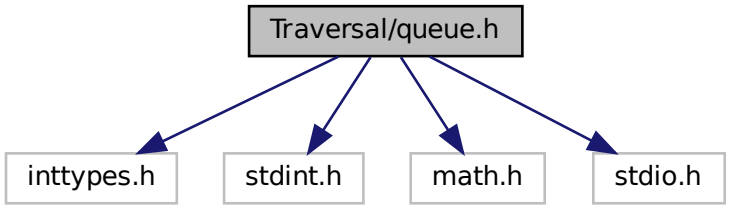
```
33     long double m;
34     long double c;
35 };
36
37 struct point {
38     long double x;
39     long double y;
40     // double;
41 };
42
52 uint8_t get_line(struct line_equation* line, struct point* a, struct point* b);
53
62 long double distance_from_line(struct line_equation* line, struct point* a);
63
72 int8_t is_next_point(struct point* a, struct point* next);
73
82 int8_t is_outside_buffer(struct line_equation* line, struct point* a);
83
95 int8_t is_possible(struct point* a, struct point* b, float time);
96
97 #endif
```

4.3 Traversal/queue.h File Reference

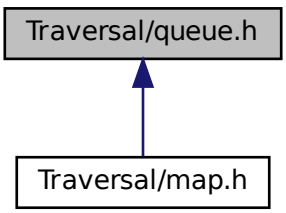
Queue system used to track the number of out-of-bonds occurences.

```
#include <inttypes.h>
#include <stdint.h>
#include <math.h>
#include <stdio.h>
```

Include dependency graph for queue.h:



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



Data Structures

- struct queue

Macros

- #define **QUEUE_LEN** 10
- #define **ON_COURSE** 0
- #define **OFF_COURSE** 1

Functions

- void print_q (struct queue *q)

  Simple helper function to print the contents of the queue.
- void enqueue (struct queue *q, int8_t inside)

  Shift all values in the queue to the left and add the new value onto the start.
- int8_t is_off_course (struct queue *q)

  Based on the percentage of out-of-bounds points determine if the robot is off course.
- void flush (struct queue *q)

  A simple helper function to set all values of the queue to 0.

4.3.1 Detailed Description

Queue system used to track the number of out-of-bonds occurences.

Author

T. Buckingham

Date

Thu Feb 16 10:50:59 2023

When a defined percentage of points are outside the area of inaccuracy the system will decide it is off course. A small number of points may be an error and so multiple are used to confirm whether the robot is off course or not.

4.3.2 Function Documentation

4.3.2.1 enqueue()

```
void enqueue (
            struct queue * q,
            int8_t inside )
```

Shift all values in the queue to the left and add the new value onto the start.

Parameters

q	The queue to process
inside	A 1 or 0, depending if the robot was outside of the boundary.

4.3.2.2 flush()

```
void flush (
            struct queue * q )
```

A simple helper function to set all values of the queue to 0.

Parameters

q	The queue to reset.

4.3.2.3 is_off_course()

```
int8_t is_off_course (
            struct queue * q )
```

Based on the percentage of out-of-bounds points determine if the robot is off course.

Parameters

q	The queue holding the stored out-of-bounds values.

Returns

Whether the robot is OFF_COURSE or ON_COURSE

4.3.2.4 print_q()

```
void print_q (
            struct queue * q )
```

Simple helper function to print the contents of the queue.

Parameters

q	The queue to iterate over.

4.4 queue.h

Go to the documentation of this file.

```
1
15 #ifndef QUEUE_H
16 #define QUEUE_H
17
18 #include <inttypes.h>
19 #include <stdint.h>
20 #include <math.h>
21 #include <stdio.h>
22
23 #define QUEUE_LEN 10
24 #define ON_COURSE  0
25 #define OFF_COURSE 1
26
27 struct queue {
28     int8_t inside[QUEUE_LEN];
29     int8_t length;
30 };
31
37 void print_q(struct queue* q);
38
46 void enqueue(struct queue* q, int8_t inside);
47
56 int8_t is_off_course(struct queue* q);
57
63 void flush(struct queue* q);
64
65 #endif
```

4.5 Traversal/test_data.h File Reference

Header file for the outputted test data.

Variables

- long double **route** [50000][2] = {0}
- long double **positions** [50000][2] = {0}
- int **route_len**
- int **positions_len**

4.5.1 Detailed Description

Header file for the outputted test data.

Author

T. Buckingham

Date

Thu Feb 16 11:07:53 2023

Test data can be defined directly, or using the output.py script, multiple files can be iterated over through file reading and writing.

4.6 test_data.h

Go to the documentation of this file.

```
1
14 #ifndef TEST_DATA_H
15 #define TEST_DATA_H
16
17 long double route[50000][2] = {0};      /* The route for the robot to follow  */
18 long double positions[50000][2] = {0};  /* Fictitious testing data, simulating the robot's recorded
       positions */
19 int route_len;
20 int positions_len;
21
22 #endif
```

Index