

My Project

Generated by Doxygen 1.8.11

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Motor Struct Reference	5
3.1.1	Member Data Documentation	5
3.1.1.1	pin1	5
3.1.1.2	pin2	5
3.1.1.3	sensor1	5
3.1.1.4	sensor2	5
3.1.1.5	speed	5
3.2	Stepper Struct Reference	5
3.2.1	Member Data Documentation	6
3.2.1.1	pin1	6
3.2.1.2	pin2	6
3.2.1.3	sensor1	6
3.2.1.4	sensor2	6
3.2.1.5	speed	6

4	File Documentation	7
4.1	motor/main.c File Reference	7
4.2	stepper/main.c File Reference	7
4.2.1	Function Documentation	7
4.2.1.1	main(int argc, char **argv)	7
4.3	motor/motor.c File Reference	7
4.3.1	Function Documentation	8
4.3.1.1	motor_init(int pin1, int pin2, int sens1, int sens2)	8
4.3.1.2	motor_run_stop(Motor m, int direction)	8
4.3.1.3	motor_run_time(Motor m, int direction, int ms)	8
4.4	motor/motor.h File Reference	9
4.4.1	Detailed Description	9
4.4.2	Macro Definition Documentation	9
4.4.2.1	BACKWARDS	9
4.4.2.2	FORWARD	9
4.4.2.3	SAMPLE_DELAY	9
4.4.3	Typedef Documentation	9
4.4.3.1	Motor	9
4.4.4	Function Documentation	9
4.4.4.1	motor_init(int pin1, int pin2, int sens1, int sens2)	9
4.4.4.2	motor_run_stop(Motor m, int direction)	10
4.4.4.3	motor_run_time(Motor m, int direction, int ms)	10
4.4.4.4	motor_set_speed(Motor *m, int speed)	10
4.5	stepper/stepper.c File Reference	10
4.5.1	Function Documentation	11
4.5.1.1	set_speed(Stepper *s, int speed)	11
4.5.1.2	step(Stepper s, int steps)	11
4.5.1.3	stepper_init(int pin1, int pin2, int sens1, int sens2)	11
4.6	stepper/stepper.h File Reference	12
4.6.1	Detailed Description	12
4.6.2	Typedef Documentation	12
4.6.2.1	Stepper	12
4.6.3	Function Documentation	12
4.6.3.1	set_speed(Stepper *s, int speed)	12
4.6.3.2	step(Stepper s, int steps)	13
4.6.3.3	stepper_init(int pin1, int pin2, int sens1, int sens2)	13
	Index	15

Chapter 1

Class Index

1.1 Class List

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

Motor	5
Stepper	5

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

motor/ main.c	7
motor/ motor.c	7
motor/ motor.h	
Motor control functions with end of track detection	9
stepper/ main.c	7
stepper/ stepper.c	10
stepper/ stepper.h	
Functions for a 2-pin bound stepper motor (1 pin per step in each direction) with end-of-track detection	12

Chapter 3

Class Documentation

3.1 Motor Struct Reference

```
#include <motor.h>
```

Public Attributes

- int [pin1](#)
- int [pin2](#)
- int [sensor1](#)
- int [sensor2](#)
- int [speed](#)

3.1.1 Member Data Documentation

3.1.1.1 int Motor::pin1

3.1.1.2 int Motor::pin2

3.1.1.3 int Motor::sensor1

3.1.1.4 int Motor::sensor2

3.1.1.5 int Motor::speed

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

- motor/[motor.h](#)

3.2 Stepper Struct Reference

```
#include <stepper.h>
```

Public Attributes

- int [pin1](#)
- int [pin2](#)
- int [sensor1](#)
- int [sensor2](#)
- int [speed](#)

3.2.1 Member Data Documentation

3.2.1.1 int `Stepper::pin1`

3.2.1.2 int `Stepper::pin2`

3.2.1.3 int `Stepper::sensor1`

3.2.1.4 int `Stepper::sensor2`

3.2.1.5 int `Stepper::speed`

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

- [stepper/stepper.h](#)

Chapter 4

File Documentation

4.1 motor/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include "motor.h"
Include dependency graph for main.c:
```

4.2 stepper/main.c File Reference

```
#include "stepper.h"
#include <stdio.h>
#include <stdlib.h>
Include dependency graph for main.c:
```

Functions

- int [main](#) (int argc, char **argv)

4.2.1 Function Documentation

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

4.3 motor/motor.c File Reference

```
#include "motor.h"
Include dependency graph for motor.c:
```

Functions

- **Motor** `motor_init` (int pin1, int pin2, int sens1, int sens2)
initializes motor with end of track sensors
- void `motor_run_stop` (**Motor** m, int direction)
runs the motor in the given direction until it hits its end-of-track sensor. Only checks for sensor 1 if run in direction 1, sensor 2 if run in direction 2
- void `motor_run_time` (**Motor** m, int direction, int ms)
runs the motor for a specified amount of time without checking for end-of-track sensor detection

4.3.1 Function Documentation

4.3.1.1 **Motor** `motor_init` (int *pin1*, int *pin2*, int *sens1*, int *sens2*)

initializes motor with end of track sensors

Parameters

<i>pin1, pin2</i>	the GPIO for running motor in directions 1,2
<i>sens1, sens2</i>	the GPIO input pin for end of track sensor 1,2

Returns

the generated motor descriptor

4.3.1.2 void `motor_run_stop` (**Motor** *m*, int *direction*)

runs the motor in the given direction until it hits its end-of-track sensor. Only checks for sensor 1 if run in direction 1, sensor 2 if run in direction 2

Parameters

<i>m</i>	an initialized motor descriptor
<i>direction</i>	the direction (BACKWARDS=2, FORWARD=1)

4.3.1.3 void `motor_run_time` (**Motor** *m*, int *direction*, int *ms*)

runs the motor for a specified amount of time without checking for end-of-track sensor detection

Parameters

<i>m</i>	an initialized motor descriptor
<i>direction</i>	the direction
<i>ms</i>	the amount of time to run the command in ms

4.4 motor/motor.h File Reference

motor control functions with end of track detection

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for motor.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [Motor](#)

Macros

- #define [FORWARD](#) 1
- #define [BACKWARDS](#) 2
- #define [SAMPLE_DELAY](#) 1

Typedefs

- typedef struct [Motor](#) [Motor](#)

Functions

- [Motor](#) [motor_init](#) (int pin1, int pin2, int sens1, int sens2)
initializes motor with end of track sensors
- void [motor_set_speed](#) ([Motor](#) *m, int speed)
sets the speed for a motor (unimplemented now)
- void [motor_run_stop](#) ([Motor](#) m, int direction)
runs the motor in the given direction until it hits its end-of-track sensor. Only checks for sensor 1 if run in direction 1, sensor 2 if run in direction 2
- void [motor_run_time](#) ([Motor](#) m, int direction, int ms)
runs the motor for a specified amount of time without checking for end-of-track sensor detection

4.4.1 Detailed Description

motor control functions with end of track detection

4.4.2 Macro Definition Documentation

4.4.2.1 #define [BACKWARDS](#) 2

4.4.2.2 #define [FORWARD](#) 1

4.4.2.3 #define [SAMPLE_DELAY](#) 1

4.4.3 Typedef Documentation

4.4.3.1 typedef struct [Motor](#) [Motor](#)

4.4.4 Function Documentation

4.4.4.1 [Motor](#) [motor_init](#) (int *pin1*, int *pin2*, int *sens1*, int *sens2*)

initializes motor with end of track sensors

Parameters

<i>pin1, pin2</i>	the GPIO for running motor in directions 1,2
<i>sens1, sens2</i>	the GPIO input pin for end of track sensor 1,2

Returns

the generated motor descriptor

4.4.4.2 void motor_run_stop (Motor *m*, int *direction*)

runs the motor in the given direction until it hits its end-of-track sensor. Only checks for sensor 1 if run in direction 1, sensor 2 if run in direction 2

Parameters

<i>m</i>	an initialized motor descriptor
<i>direction</i>	the direction (BACKWARDS=2, FORWARD=1)

4.4.4.3 void motor_run_time (Motor *m*, int *direction*, int *ms*)

runs the motor for a specified amount of time without checking for end-of-track sensor detection

Parameters

<i>m</i>	an initialized motor descriptor
<i>direction</i>	the direction
<i>ms</i>	the amount of time to run the command in ms

4.4.4.4 void motor_set_speed (Motor * *m*, int *speed*)

sets the speed for a motor (unimplemented now)

Parameters

<i>m</i>	a pointer to an initialized motor structure
<i>speed</i>	the new speed to be set

4.5 stepper/stepper.c File Reference

```
#include "stepper.h"
```

Include dependency graph for stepper.c:

Functions

- [Stepper stepper_init](#) (int pin1, int pin2, int sens1, int sens2)
initializes a stepper motor descriptor structure with output pins and sensor input pins
- void [set_speed](#) ([Stepper](#) *s, int speed)
sets the running speed for the stepper motor
- int [step](#) ([Stepper](#) s, int steps)
step the motor by the given amount (positive and negative) returns the number of steps actually done (stops at sensor cut)

4.5.1 Function Documentation

4.5.1.1 void set_speed ([Stepper](#) * s, int speed)

sets the running speed for the stepper motor

Parameters

s	an initialized stepper descriptor
speed	the new speed

4.5.1.2 int step ([Stepper](#) s, int steps)

step the motor by the given amount (positive and negative) returns the number of steps actually done (stops at sensor cut)

Parameters

s	an initialized stepper descriptor
steps	the number of steps performed

Returns

the true number of steps performed

4.5.1.3 [Stepper stepper_init](#) (int pin1, int pin2, int sens1, int sens2)

initializes a stepper motor descriptor structure with output pins and sensor input pins

Parameters

pin1,pin2	the pins for the stepper to step
sens1,sens2	the pins for the end of track sensors

4.6 stepper/stepper.h File Reference

contains functions for a 2-pin bound stepper motor (1 pin per step in each direction) with end-of-track detection

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for stepper.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [Stepper](#)

Typedefs

- typedef struct [Stepper](#) [Stepper](#)

Functions

- [Stepper stepper_init](#) (int pin1, int pin2, int sens1, int sens2)
initializes a stepper motor descriptor structure with output pins and sensor input pins
- void [set_speed](#) ([Stepper](#) *s, int speed)
sets the running speed for the stepper motor
- int [step](#) ([Stepper](#) s, int steps)
step the motor by the given amount (positive and negative) returns the number of steps actually done (stops at sensor cut)

4.6.1 Detailed Description

contains functions for a 2-pin bound stepper motor (1 pin per step in each direction) with end-of-track detection

4.6.2 Typedef Documentation

4.6.2.1 typedef struct [Stepper](#) [Stepper](#)

4.6.3 Function Documentation

4.6.3.1 void [set_speed](#) ([Stepper](#) * s, int *speed*)

sets the running speed for the stepper motor

Parameters

<i>s</i>	an initialized stepper descriptor
<i>speed</i>	the new speed

4.6.3.2 int step (Stepper *s*, int *steps*)

step the motor by the given amount (positive and negative) returns the number of steps actually done (stops at sensor cut)

Parameters

<i>s</i>	an initialized stepper descriptor
<i>steps</i>	the number of steps performed

Returns

the true number of steps performed

4.6.3.3 Stepper stepper_init (int *pin1*, int *pin2*, int *sens1*, int *sens2*)

initializes a stepper motor descriptor structure with output pins and sensor input pins

Parameters

<i>pin1, pin2</i>	the pins for the stepper to step
<i>sens1, sens2</i>	the pins for the end of track sensors

Index

BACKWARDS

motor.h, [9](#)

FORWARD

motor.h, [9](#)

main

stepper/main.c, [7](#)

Motor, [5](#)

motor.h, [9](#)

pin1, [5](#)

pin2, [5](#)

sensor1, [5](#)

sensor2, [5](#)

speed, [5](#)

motor.c

motor_init, [8](#)

motor_run_stop, [8](#)

motor_run_time, [8](#)

motor.h

BACKWARDS, [9](#)

FORWARD, [9](#)

Motor, [9](#)

motor_init, [9](#)

motor_run_stop, [10](#)

motor_run_time, [10](#)

motor_set_speed, [10](#)

SAMPLE_DELAY, [9](#)

motor/main.c, [7](#)

motor/motor.c, [7](#)

motor/motor.h, [9](#)

motor_init

motor.c, [8](#)

motor.h, [9](#)

motor_run_stop

motor.c, [8](#)

motor.h, [10](#)

motor_run_time

motor.c, [8](#)

motor.h, [10](#)

motor_set_speed

motor.h, [10](#)

pin1

Motor, [5](#)

Stepper, [6](#)

pin2

Motor, [5](#)

Stepper, [6](#)

SAMPLE_DELAY

motor.h, [9](#)

sensor1

Motor, [5](#)

Stepper, [6](#)

sensor2

Motor, [5](#)

Stepper, [6](#)

set_speed

stepper.c, [11](#)

stepper.h, [12](#)

speed

Motor, [5](#)

Stepper, [6](#)

step

stepper.c, [11](#)

stepper.h, [13](#)

Stepper, [5](#)

pin1, [6](#)

pin2, [6](#)

sensor1, [6](#)

sensor2, [6](#)

speed, [6](#)

stepper.h, [12](#)

stepper.c

set_speed, [11](#)

step, [11](#)

stepper_init, [11](#)

stepper.h

set_speed, [12](#)

step, [13](#)

Stepper, [12](#)

stepper_init, [13](#)

stepper/main.c, [7](#)

main, [7](#)

stepper/stepper.c, [10](#)

stepper/stepper.h, [12](#)

stepper_init

stepper.c, [11](#)

stepper.h, [13](#)