My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

pin1,pin2	the GPIO for running motor in directions 1,2
sens1,sens2	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

m	an initialized motor descriptor
direction	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

m	an initialized motor descriptor
direction	the direction
ms	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

pin1,pin2	the GPIO for running motor in directions 1,2
sens1,sens2	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

m	an initialized motor descriptor
direction	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

m	an initialized motor descriptor
direction	the direction
ms	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

m	a pointer to an initialized motor structure
speed	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

s	an initialized stepper descriptor
speed	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

s	an initialized stepper descriptor
steps	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

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

Index