Heisprosjekt TTK4235

Generated by Doxygen 1.8.11

Contents

Chapter 1

HeisProsjekt

Heisprosjekt i C

How to build:

Using docker:

Install docker and run:

```
docker build .
```

to build and test code

Using CMake and make

Install CMake:

```
sudo apt-get install cmake
```

Create build directory and change directory

```
mkdir build
cd build
```

Run CMake with parent directory

```
cmake ..
```

Run make on generated makefile

```
make
```

Run executable

```
./heisprosjekt
```

Using build script

In project folder, run either build.sh or build_test.sh

```
./build.sh
```

If the scripts won't start, make it an executable

```
chmod +x build.sh
```

Run the built binary

```
./heisprosjekt
```

How to build parent image

```
docker build . -f docker/system/Dockerfile -t YOUR_TAG_HERE
docker push YOUR_TAG_HERE
```

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 state_data_t Struct Reference

Struct to be passed in, containing useful data.

```
#include <fsm.h>
```

Public Attributes

- motor_direction_e motor_direction

  Current motor direction.
- motor_running_e motor_running

  Motor running state.
- bool emergency_button_pressed

  Is emergency button pressed?
- int target_floor

  Elevator target floor (current order)
- int current_floor

  Current floor (-1 if in between floors)
- int last_floor

  Last visited floor.

4.1.1 Detailed Description

Struct to be passed in, containing useful data.

4.1.2 Member Data Documentation

4.1.2.1 int state_data_t::current_floor

Current floor (-1 if in between floors)

4.1.2.2 bool state_data_t::emergency_button_pressed

Is emergency button pressed?

4.1.2.3 int state_data_t::last_floor

Last visited floor.

4.1.2.4 motor_direction_e state_data_t::motor_direction

Current motor direction.

4.1.2.5 motor_running_e state_data_t::motor_running

Motor running state.

4.1.2.6 int state_data_t::target_floor

Elevator target floor (current order)

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

- include/fsm.h

4.2 timer_t Struct Reference

Timer struct.

```
#include <timer_driver.h>
```

Public Attributes

- clock_t start_time
- clock_t duration_ms

4.2.1 Detailed Description

Timer struct.

4.2.2 Member Data Documentation

4.2.2.1 clock_t timer_t::duration_ms

4.2.2.2 clock_t timer_t::start_time

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

- include/timer_driver.h

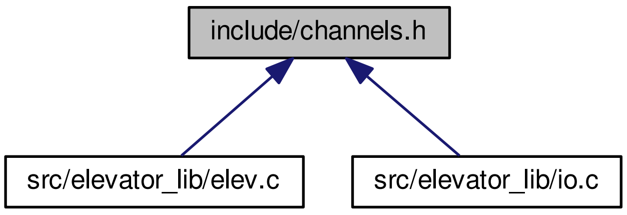
Chapter 5

File Documentation

5.1 CMakeLists.txt File Reference

5.2 include/channels.h File Reference

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



Macros

- #define PORT4 3
- #define OBSTRUCTION (0x300+23)
- #define STOP (0x300+22)
- #define BUTTON_COMMAND1 (0x300+21)
- #define BUTTON_COMMAND2 (0x300+20)
- #define BUTTON_COMMAND3 (0x300+19)
- #define BUTTON_COMMAND4 (0x300+18)
- #define BUTTON_UP1 (0x300+17)
- #define BUTTON_UP2 (0x300+16)
- #define PORT1 2
- #define BUTTON_DOWN2 (0x200+0)
- #define BUTTON_UP3 (0x200+1)
- #define BUTTON_DOWN3 (0x200+2)

- #define BUTTON_DOWN4 (0x200+3)
- #define SENSOR_FLOOR1 (0x200+4)
- #define SENSOR_FLOOR2 (0x200+5)
- #define SENSOR_FLOOR3 (0x200+6)
- #define SENSOR_FLOOR4 (0x200+7)
- #define PORT3 3
- #define MOTORDIR (0x300+15)
- #define LIGHT_STOP (0x300+14)
- #define LIGHT_COMMAND1 (0x300+13)
- #define LIGHT_COMMAND2 (0x300+12)
- #define LIGHT_COMMAND3 (0x300+11)
- #define LIGHT_COMMAND4 (0x300+10)
- #define LIGHT_UP1 (0x300+9)
- #define LIGHT_UP2 (0x300+8)
- #define PORT2 3
- #define LIGHT_DOWN2 (0x300+7)
- #define LIGHT_UP3 (0x300+6)
- #define LIGHT_DOWN3 (0x300+5)
- #define LIGHT_DOWN4 (0x300+4)
- #define LIGHT_DOOR_OPEN (0x300+3)
- #define LIGHT_FLOOR_IND2 (0x300+1)
- #define LIGHT_FLOOR_IND1 (0x300+0)
- #define PORT0 1
- #define MOTOR (0x100+0)
- #define BUTTON_DOWN1 -1
- #define BUTTON_UP4 -1
- #define LIGHT_DOWN1 -1
- #define LIGHT_UP4 -1

5.2.1 Macro Definition Documentation

5.2.1.1 #define BUTTON_COMMAND1 (0x300+21)

5.2.1.2 #define BUTTON_COMMAND2 (0x300+20)

5.2.1.3 #define BUTTON_COMMAND3 (0x300+19)

5.2.1.4 #define BUTTON_COMMAND4 (0x300+18)

5.2.1.5 #define BUTTON_DOWN1 -1

5.2.1.6 #define BUTTON_DOWN2 (0x200+0)

5.2.1.7 #define BUTTON_DOWN3 (0x200+2)

5.2.1.8 #define BUTTON_DOWN4 (0x200+3)

5.2.1.9 #define BUTTON_UP1 (0x300+17)

5.2.1.10 #define BUTTON_UP2 (0x300+16)

5.2.1.11 #define BUTTON_UP3 (0x200+1)

5.2.1.12 #define BUTTON_UP4 -1

5.2.1.13 #define LIGHT_COMMAND1 (0x300+13)

5.2.1.14 #define LIGHT_COMMAND2 (0x300+12)

5.2.1.15 #define LIGHT_COMMAND3 (0x300+11)

5.2.1.16 #define LIGHT_COMMAND4 (0x300+10)

5.2.1.17 #define LIGHT_DOOR_OPEN (0x300+3)

5.2.1.18 #define LIGHT_DOWN1 -1

5.2.1.19 #define LIGHT_DOWN2 (0x300+7)

5.2.1.20 #define LIGHT_DOWN3 (0x300+5)

5.2.1.21 #define LIGHT_DOWN4 (0x300+4)

5.2.1.22 #define LIGHT_FLOOR_IND1 (0x300+0)

5.2.1.23 #define LIGHT_FLOOR_IND2 (0x300+1)

5.2.1.24 #define LIGHT_STOP (0x300+14)

5.2.1.25 #define LIGHT_UP1 (0x300+9)

5.2.1.26 #define LIGHT_UP2 (0x300+8)

5.2.1.27 #define LIGHT_UP3 (0x300+6)

5.2.1.28 #define LIGHT_UP4 -1

5.2.1.29 #define MOTOR (0x100+0)

5.2.1.30 #define MOTORDIR (0x300+15)

5.2.1.31 #define OBSTRUCTION (0x300+23)

5.2.1.32 #define PORT0 1

5.2.1.33 #define PORT1 2

5.2.1.34 #define PORT2 3

5.2.1.35 #define PORT3 3

5.2.1.36 #define PORT4 3

5.2.1.37 #define SENSOR_FLOOR1 (0x200+4)

5.2.1.38 #define SENSOR_FLOOR2 (0x200+5)

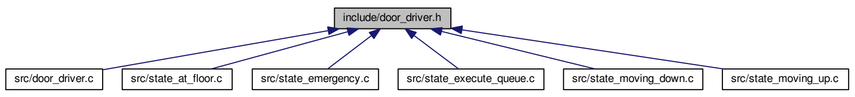
5.2.1.39 #define SENSOR_FLOOR3 (0x200+6)

5.2.1.40 #define SENSOR_FLOOR4 (0x200+7)

5.2.1.41 #define STOP (0x300+22)

5.3 include/door_driver.h File Reference

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



Enumerations

- enum door_state_e { DOOR_CLOSED = 0, DOOR_OPEN = 1 }

Functions

- void open_door (void)
- void close_door (void)
- door_state_e is_door_open (void)

5.3.1 Enumeration Type Documentation

5.3.1.1 enum door_state_e

Enumerator

DOOR_CLOSED

DOOR_OPEN

5.3.2 Function Documentation

5.3.2.1 void close_door (void)

Closes door

5.3.2.2 door_state_e is_door_open (void)

Checks if door is open

Returns

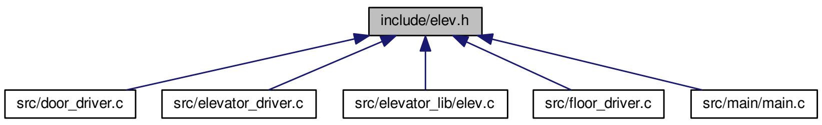
DOOR_OPEN if door is open
DOOR_CLOSED if door is closed

5.3.2.3 void open_door (void)

Opens door

5.4 include/elev.h File Reference

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



Macros

- #define N_FLOORS 4

Typedefs

- typedef enum tag_elev_motor_direction elev_motor_direction_t
- typedef enum tag_elev_lamp_type elev_button_type_t

Enumerations

- enum tag_elev_motor_direction { DIRN_DOWN = -1, DIRN_STOP = 0, DIRN_UP = 1 }
- enum tag_elev_lamp_type { BUTTON_CALL_UP = 0, BUTTON_CALL_DOWN = 1, BUTTON_COMMAND = 2 }

Functions

- int elev_init (void)
- void elev_set_motor_direction (elev_motor_direction_t dirn)
- void elev_set_door_open_lamp (int value)
- int elev_get_obstruction_signal (void)
- int elev_get_stop_signal (void)
- void elev_set_stop_lamp (int value)
- int elev_get_floor_sensor_signal (void)
- void elev_set_floor_indicator (int floor)
- int elev_get_button_signal (elev_button_type_t button, int floor)
- void elev_set_button_lamp (elev_button_type_t button, int floor, int value)

5.4.1 Macro Definition Documentation

5.4.1.1 #define N_FLOORS 4

5.4.2 Typedef Documentation

5.4.2.1 typedef enum **tag_elev_lamp_type elev_button_type_t**

Button types for function elev_set_button_lamp() and elev_get_button().

5.4.2.2 typedef enum **tag_elev_motor_direction elev_motor_direction_t**

Motor direction for function elev_set_motor_direction().

5.4.3 Enumeration Type Documentation

5.4.3.1 enum **tag_elev_lamp_type**

Button types for function elev_set_button_lamp() and elev_get_button().

Enumerator

BUTTON_CALL_UP
BUTTON_CALL_DOWN
BUTTON_COMMAND

5.4.3.2 enum **tag_elev_motor_direction**

Motor direction for function elev_set_motor_direction().

Enumerator

DIRN_DOWN
DIRN_STOP
DIRN_UP

5.4.4 Function Documentation

5.4.4.1 int elev_get_button_signal (**elev_button_type_t** *button*, int *floor*)

Gets a button signal.

Parameters

button	Which button type to check. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside the elevator).
floor	Which floor to check button. Must be 0-3.

Returns

0 if button is not pushed. 1 if button is pushed.

5.4.4.2 int elev_get_floor_sensor_signal (void)

Get floor sensor signal.

Returns

-1 if elevator is not on a floor. 0-3 if elevator is on floor. 0 is ground floor, 3 is top floor.

5.4.4.3 int elev_get_obstruction_signal (void)

Get signal from obstruction switch.

Returns

1 if obstruction is enabled. 0 if not.

5.4.4.4 int elev_get_stop_signal (void)

Get signal from stop button.

Returns

1 if stop button is pushed, 0 if not.

5.4.4.5 int elev_init (void)

Initialize elevator.

Returns

Non-zero on success, 0 on failure.

5.4.4.6 void elev_set_button_lamp (elev_button_type_t *button,* int *floor,* int *value*)

Set a button lamp.

Parameters

lamp	Which type of lamp to set. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside" the elevator).
floor	Floor of lamp to set. Must be 0-3
value	Non-zero value turns lamp on, 0 turns lamp off.

5.4.4.7 void elev_set_door_open_lamp (int *value*)

Turn door-open lamp on or off.

Parameters

value	Non-zero value turns lamp on, 0 turns lamp off.

5.4.4.8 void elev_set_floor_indicator (int *floor*)

Set floor indicator lamp for a given floor.

Parameters

floor	Which floor lamp to turn on. Other floor lamps are turned off.

5.4.4.9 void elev_set_motor_direction (elev_motor_direction_t *dirn*)

Sets the motor direction of the elevator.

Parameters

dirn	New direction of the elevator.

5.4.4.10 void elev_set_stop_lamp (int *value*)

Turn stop lamp on or off.

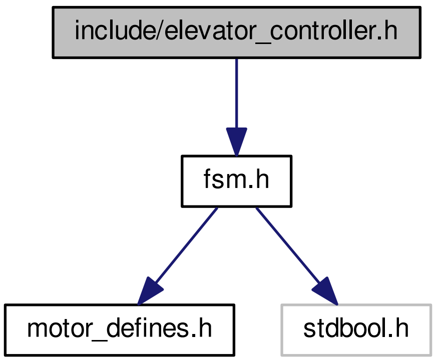
Parameters

value	Non-zero value turns lamp on, 0 turns lamp off.

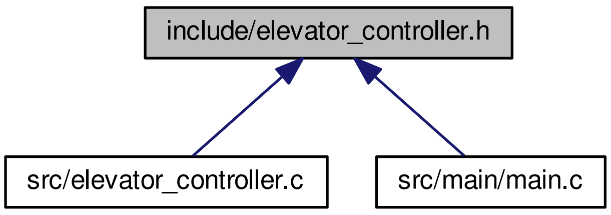
5.5 include/elevator_controller.h File Reference

```
#include "fsm.h"
```

Include dependency graph for elevator_controller.h:



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



Functions

- void elevator_controller_loop_once ()

  Run current elevator state once.

5.5.1 Function Documentation

5.5.1.1 void elevator_controller_loop_once ( )

Run current elevator state once.

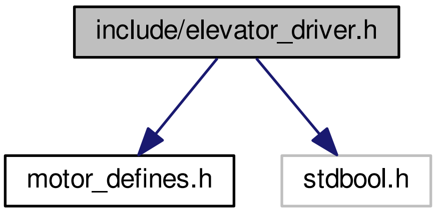
Get next state function

Run current state

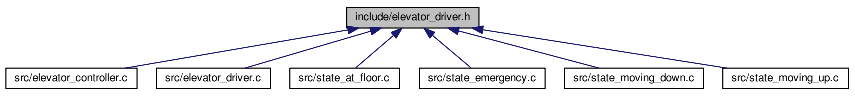
5.6 include/elevator_driver.h File Reference

```
#include "motor_defines.h"
#include "stdbool.h"
```

Include dependency graph for elevator_driver.h:



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



Functions

- void start_motor (void)
- void stop_motor (void)
- motor_direction_e get_motor_direction (void)
- void set_motor_direction (motor_direction_e dir)
- motor_running_e is_motor_running (void)
- bool is_emergency_button_pressed (void)
- void clear_elevator_light (int floor)
- void update_elevator_driver (bool init_complete)

5.6.1 Function Documentation

5.6.1.1 void clear_elevator_light (int *floor*)

Clears the elevator light of the desired floor

Parameters

floor	is the desired floor

5.6.1.2 motor_direction_e get_motor_direction (void)

Returns motor direction

Returns

MOTOR_DIRECTION_UP if direction is up
MOTOR_DIRECTION_DOWN if direction is down

5.6.1.3 bool is_emergency_button_pressed (void)

Checks if the emeregency button is pressed

Returns

EMERGENCY_NOT_PRESSED if button is not pressed
EMERGENCY_PRESSED if button is pressed

5.6.1.4 motor_running_e is_motor_running (void)

Checks if the elevator is moving

Returns

MOTOR_RUNNING if elevator is moving
MOTOR_NOT_RUNNING if elevator is still

5.6.1.5 void set_motor_direction (motor_direction_e *dir*)

Set motor direction

Parameters

dir	is the desired direction of the motor

5.6.1.6 void start_motor (void)

Starts motor

5.6.1.7 void stop_motor (void)

Stops motor

5.6.1.8 void update_elevator_driver (bool *init_complete*)

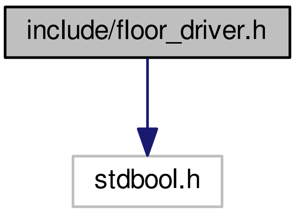
Updates the module

Parameters

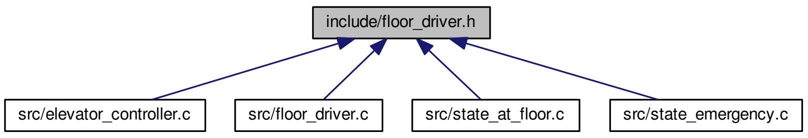
init_complete	Init is completed

5.7 include/floor_driver.h File Reference

`#include <stdbool.h>`
Include dependency graph for floor_driver.h:



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



Functions

- void clear_floor_light (int floor)
- void update_floor_driver (bool init_complete)
- int get_current_floor (void)
- void set_floor_indicator (int floor)

5.7.1 Function Documentation

5.7.1.1 void clear_floor_light (int *floor*)

Clear light at floor

Parameters

floor	Which floor light to clear

5.7.1.2 int get_current_floor (void)

Check if elevator is at a floor

Returns

current floor if at a floor, -1 if between floors

5.7.1.3 void set_floor_indicator (int *floor*)

Parameters

desired	floor

5.7.1.4 void update_floor_driver (bool *init_complete*)

Floor driver main function

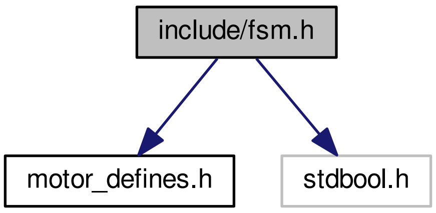
Parameters

init_complete	Init is completed

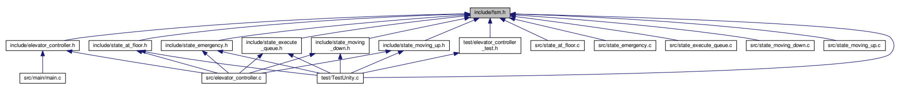
5.8 include/fsm.h File Reference

```
#include "motor_defines.h"
#include "stdbool.h"
```

Include dependency graph for fsm.h:



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


include/fsm.h
include/elevator_controller.h
include/state_at_floor.h
include/state_emergency.h
include/state_execute_queue.h
include/state_moving_down.h
include/state_moving_up.h
test/elevator_controller_test.h
src/state_at_floor.c
src/state_emergency.c
src/state_execute_queue.c
src/state_moving_down.c
src/state_moving_up.c
src/main/main.c
src/elevator_controller.c
test/TestUnity.c

Classes

- struct state_data_t

    Struct to be passed in, containing useful data.

Typedefs

- typedef fsm_state_e(∗const fsm_state_func) (const state_data_t ∗)

    FSM state function pointer type.

Enumerations

- enum fsm_state_e {
  STATE_MOVING_UP, STATE_MOVING_DOWN, STATE_EMERGENCY, STATE_EXECUTE_QUEUE,
  STATE_AT_FLOOR, FSM_NUM_STATES }

    FSM states enum.

5.8.1 Typedef Documentation

5.8.1.1 typedef **fsm_state_e**(∗ const fsm_state_func) (const **state_data_t** ∗)

FSM state function pointer type.

5.8.2 Enumeration Type Documentation

5.8.2.1 enum **fsm_state_e**

FSM states enum.

Enumerator

STATE_MOVING_UP

STATE_MOVING_DOWN

STATE_EMERGENCY

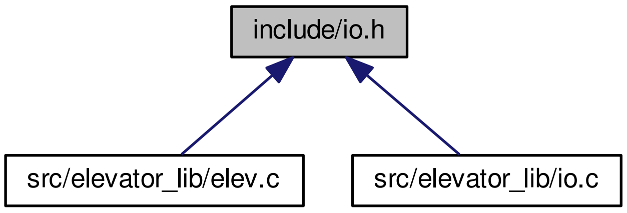
STATE_EXECUTE_QUEUE

STATE_AT_FLOOR

FSM_NUM_STATES

5.9 include/io.h File Reference

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



Functions

- int io_init ()
- void io_set_bit (int channel)
- void io_clear_bit (int channel)
- void io_write_analog (int channel, int value)
- int io_read_bit (int channel)
- int io_read_analog (int channel)

5.9.1 Function Documentation

5.9.1.1 void io_clear_bit (int *channel*)

Clears a digital channel bit.

Parameters

channel	Channel bit to set.

5.9.1.2 int io_init ( )

Initialize libComedi in "Sanntidssalen"

Returns

Non-zero on success and 0 on failure

5.9.1.3 int io_read_analog (int *channel*)

Reads a bit value from an analog channel.

Parameters

channel	Channel to read from.

Returns

Value read.

5.9.1.4 int io_read_bit (int *channel*)

Reads a bit value from a digital channel.

Parameters

channel	Channel to read from.

Returns

Value read.

5.9.1.5 void io_set_bit (int *channel*)

Sets a digital channel bit.

Parameters

channel	Channel bit to set.

5.9.1.6 void io_write_analog (int *channel,* int *value*)

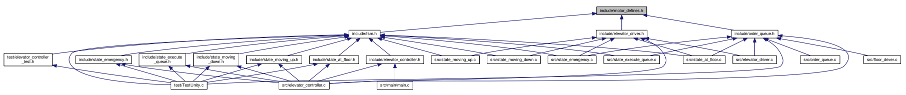
Writes a value to an analog channel.

Parameters

channel	Channel to write to.
value	Value to write.

5.10 include/motor_defines.h File Reference

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



Enumerations

- enum motor_direction_e { MOTOR_DIRECTION_DOWN = 0, MOTOR_DIRECTION_UP = 1 }
- enum motor_running_e { MOTOR_NOT_RUNNING = 0, MOTOR_RUNNING = 1 }

5.10.1 Enumeration Type Documentation

5.10.1.1 enum motor_direction_e

Enumerator

MOTOR_DIRECTION_DOWN

MOTOR_DIRECTION_UP

5.10.1.2 enum motor_running_e

Enumerator

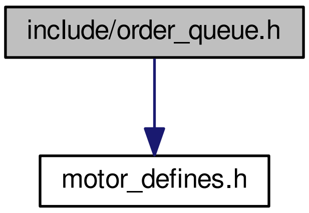
MOTOR_NOT_RUNNING

MOTOR_RUNNING

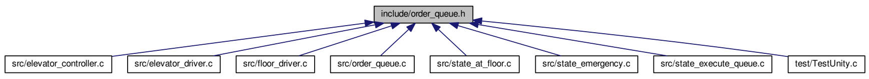
5.11 include/order_queue.h File Reference

```
#include "motor_defines.h"
```

Include dependency graph for order_queue.h:



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



Functions

- void add_to_order_queue_up (int floor)
- void add_to_order_queue_down (int floor)
- void empty_queue (void)

    Empty all queues.
- int get_next_order (int current_floor, motor_direction_e dir)
- void add_to_order_queue_dest (int floor)
- void clear_order_in_queue (int floor)

5.11.1 Function Documentation

5.11.1.1 void add_to_order_queue_dest (int *floor*)

Add down order from floor

Parameters

floor	- Which floor ordered from

5.11.1.2 void add_to_order_queue_down (int *floor*)

Add down order from floor

Parameters

floor	- Which floor ordered from

5.11.1.3 void add_to_order_queue_up (int *floor*)

Add up order from floor

Parameters

floor	- Which floor ordered from

5.11.1.4 void clear_order_in_queue (int *floor*)

Clear all orders from given floor

Parameters

floor	Floor to clear

5.11.1.5 void empty_queue (void)

Empty all queues.

5.11.1.6 int get_next_order (int *current_floor*, motor_direction_e *dir*)

Get next order from queues

Parameters

current_floor	
dir	motor direction

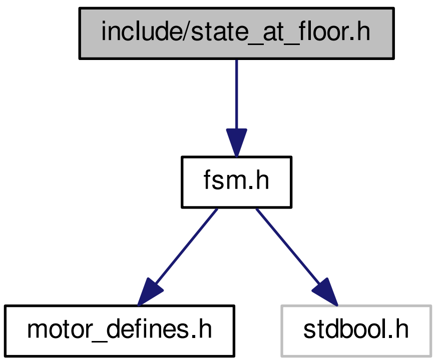
Returns

next floor to stop, -1 if no orders in queue

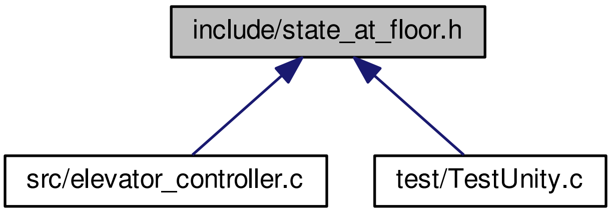
5.12 include/state_at_floor.h File Reference

```
#include "fsm.h"
```

Include dependency graph for state_at_floor.h:



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



Functions

- fsm_state_e state_at_floor_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_at_floor_do (const state_data_t ∗state_data_p)

5.12.1 Function Documentation

5.12.1.1 fsm_state_e state_at_floor_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←_p	Current system state

Returns

Next state

5.12.1.2 fsm_state_e state_at_floor_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

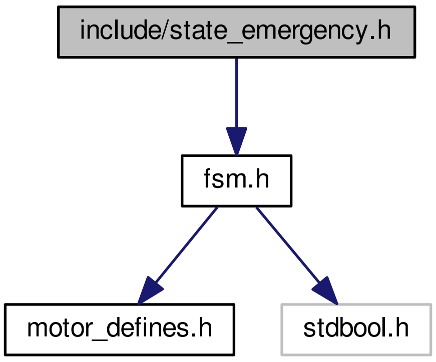
Returns

Next state

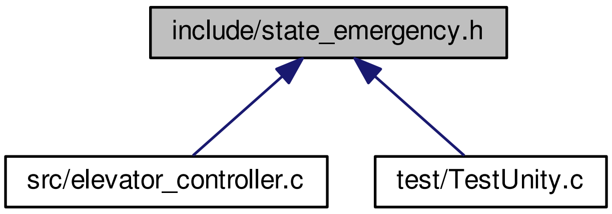
5.13 include/state_emergency.h File Reference

```
#include "fsm.h"
```

Include dependency graph for state_emergency.h:



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



Functions

- fsm_state_e state_emergency_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_emergency_do (const state_data_t ∗state_data_p)

5.13.1 Function Documentation

5.13.1.1 fsm_state_e state_emergency_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←↩ _p	Current system state

Returns

Next state

5.13.1.2 fsm_state_e state_emergency_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

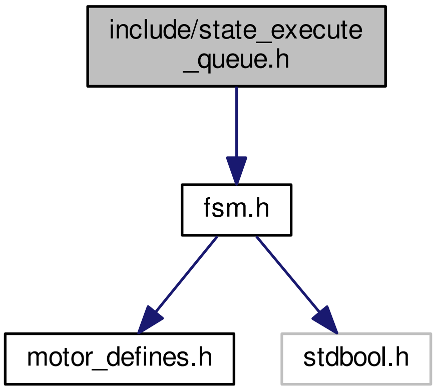
state_data←↩ _p	Current system state

Returns

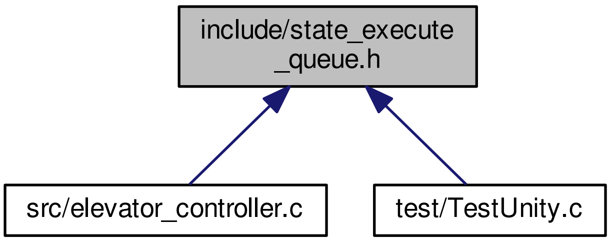
Next state

5.14 include/state_execute_queue.h File Reference

```
#include "fsm.h"
```
Include dependency graph for state_execute_queue.h:



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



Functions

- fsm_state_e state_execute_queue_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_execute_queue_do (const state_data_t ∗state_data_p)

5.14.1 Function Documentation

5.14.1.1 fsm_state_e state_execute_queue_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data↩ _p	Current system state

Returns

Next state

5.14.1.2 fsm_state_e state_execute_queue_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

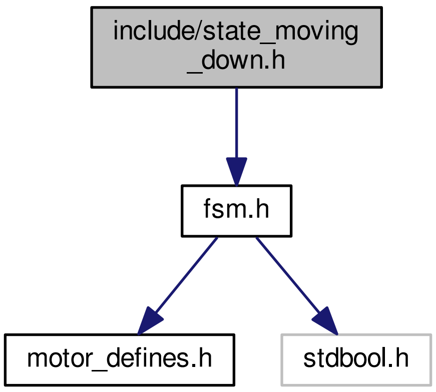
state_data↩ _p	Current system state

Returns

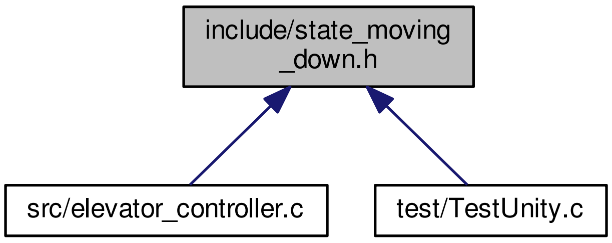
Next state

5.15 include/state_moving_down.h File Reference

`#include "fsm.h"`
Include dependency graph for state_moving_down.h:



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



Functions

- fsm_state_e state_moving_down_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_moving_down_do (const state_data_t ∗state_data_p)

5.15.1 Function Documentation

5.15.1.1 fsm_state_e state_moving_down_do (const state_data_t * *state_data_p*)

State do

Parameters

state_data←↩ _p	Current system state

Returns

Next state

5.15.1.2 fsm_state_e state_moving_down_entry (const state_data_t * *state_data_p*)

State entry

Parameters

state_data←↩ _p	Current system state

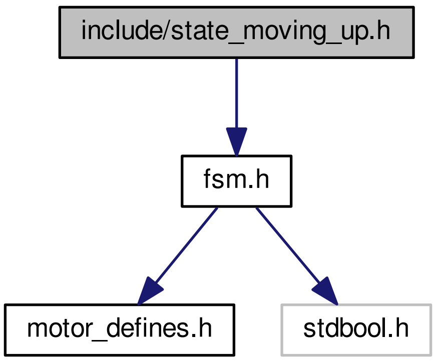
Returns

Next state

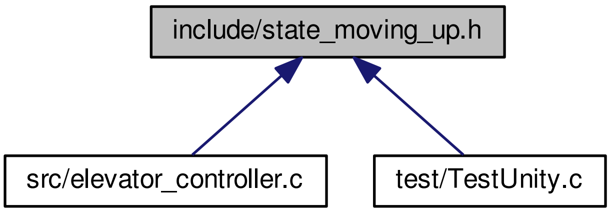
5.16 include/state_moving_up.h File Reference

```
#include "fsm.h"
```

Include dependency graph for state_moving_up.h:



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



Functions

- fsm_state_e state_moving_up_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_moving_up_do (const state_data_t ∗state_data_p)

5.16.1 Function Documentation

5.16.1.1 fsm_state_e state_moving_up_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←_p	Current system state

Returns

Next state

5.16.1.2 fsm_state_e state_moving_up_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

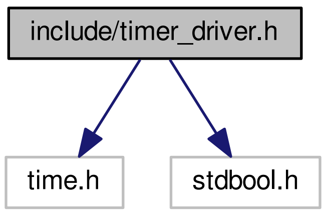
Returns

Next state

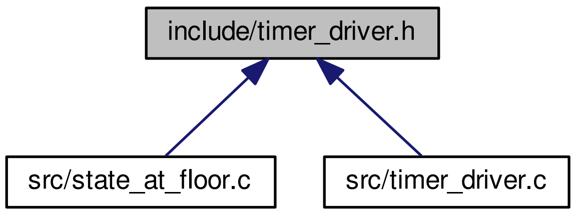
5.17 include/timer_driver.h File Reference

```
#include <time.h>
#include <stdbool.h>
```

Include dependency graph for timer_driver.h:



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



Classes

- struct timer_t

    Timer struct.

Functions

- timer_t create_ms_timer (unsigned int d_ms)
- bool timer_has_elapsed (timer_t timer)

5.17.1 Function Documentation

5.17.1.1 timer_t create_ms_timer (unsigned int *d_ms*)

Create timer

Parameters

d_ms	Timer duration in milliseconds

Returns

Timer struct

5.17.1.2 bool timer_has_elapsed (timer_t *timer*)

Check if timer has elapsed

Parameters

timer	Timer to check

Returns

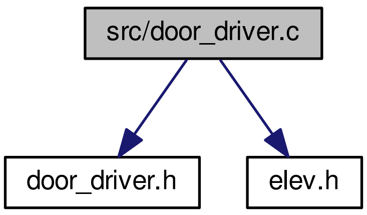
Has timer elapsed or not

5.18 README.md File Reference

5.19 src/door_driver.c File Reference

```
#include <door_driver.h>
#include "elev.h"
```

Include dependency graph for door_driver.c:



Functions

- door_state_e is_door_open (void)
- void open_door (void)
- void close_door (void)

Variables

- door_state_e door_state

5.19.1 Function Documentation

5.19.1.1 void close_door (void)

Closes door

5.19.1.2 door_state_e is_door_open (void)

Checks if door is open

Returns

DOOR_OPEN if door is open
DOOR_CLOSED if door is closed

5.19.1.3 void open_door (void)

Opens door

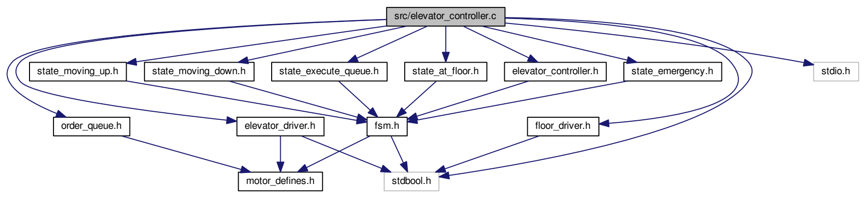
5.19.2 Variable Documentation

5.19.2.1 door_state_e door_state

5.20 src/elevator_controller.c File Reference

```
#include "elevator_controller.h"
#include <stdio.h>
#include "elevator_driver.h"
#include "floor_driver.h"
#include "order_queue.h"
#include <stdbool.h>
#include "state_emergency.h"
#include "state_moving_up.h"
#include "state_moving_down.h"
#include "state_execute_queue.h"
#include "state_at_floor.h"
```

Include dependency graph for elevator_controller.c:



Functions

- void emergency_stop (void)
- void elevator_controller_loop_once ()

  Run current elevator state once.

Variables

- fsm_state_e current_state = STATE_EXECUTE_QUEUE

  Current running state.
- fsm_state_e next_state = STATE_EXECUTE_QUEUE

  Next desired state.
- int last_floor = -1

  Last visited floor.
- bool init_complete = false

  Init is completed.
- fsm_state_func state_table [FSM_NUM_STATES][FSM_NUM_STATES]

  Contains all state functions and transition functions.

5.20.1 Function Documentation

5.20.1.1 void elevator_controller_loop_once ( )

Run current elevator state once.

Get next state function

Run current state

5.20.1.2 void emergency_stop (void)

5.20.2 Variable Documentation

5.20.2.1 fsm_state_e current_state = STATE_EXECUTE_QUEUE

Current running state.

5.20.2.2 bool init_complete = false

Init is completed.

5.20.2.3 int last_floor = -1

Last visited floor.

5.20.2.4 fsm_state_e next_state = STATE_EXECUTE_QUEUE

Next desired state.

5.20.2.5 fsm_state_func state_table[FSM_NUM_STATES][FSM_NUM_STATES]

Initial value:

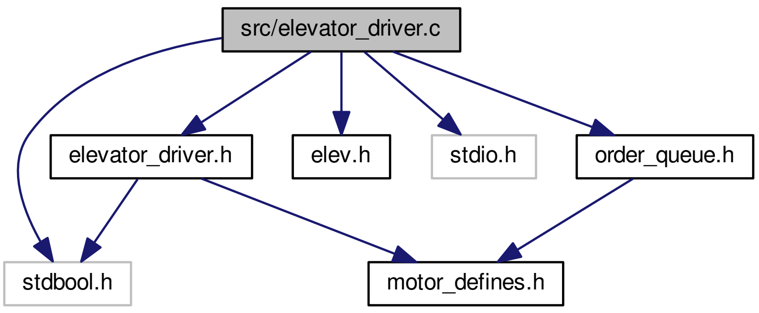
```
=
{

    { state_moving_up_do    , state_moving_down_entry,
      state_emergency_entry, state_execute_queue_entry,
      state_at_floor_entry },
    { state_moving_up_entry, state_moving_down_do    ,
      state_emergency_entry, state_execute_queue_entry,
      state_at_floor_entry },
    { state_moving_up_entry, state_moving_down_entry,
      state_emergency_do    , state_execute_queue_entry,
      state_at_floor_entry },
    { state_moving_up_entry, state_moving_down_entry,
      state_emergency_entry, state_execute_queue_do    ,
      state_at_floor_entry },
    { state_moving_up_entry, state_moving_down_entry,
      state_emergency_entry, state_execute_queue_entry,
      state_at_floor_do     }
}
```

Contains all state functions and transition functions.

5.21 src/elevator_driver.c File Reference

```
#include <elevator_driver.h>
#include "elev.h"
#include <stdbool.h>
#include <stdio.h>
#include "order_queue.h"
```

Include dependency graph for elevator_driver.c:



Macros

- #define NUM_FLOORS 4

Functions

- void start_motor (void)
- void stop_motor (void)
- motor_direction_e get_motor_direction (void)
- void set_motor_direction (motor_direction_e dir)
- motor_running_e is_motor_running (void)
- bool is_emergency_button_pressed (void)
- void clear_elevator_light (int floor)
- void set_elevator_light (int floor)
- void update_elevator_driver (bool init_complete)

Variables

- bool elevator_lights [NUM_FLOORS] = { false, false, false, false}
- motor_direction_e current_motor_direction = MOTOR_DIRECTION_UP
- motor_running_e motor_running_status = MOTOR_NOT_RUNNING

5.21.1 Macro Definition Documentation

5.21.1.1 #define NUM_FLOORS 4

5.21.2 Function Documentation

5.21.2.1 void clear_elevator_light (int *floor*)

Clears the elevator light of the desired floor

Parameters

floor	is the desired floor

5.21.2.2 motor_direction_e get_motor_direction (void)

Returns motor direction

Returns

MOTOR_DIRECTION_UP if direction is up
MOTOR_DIRECTION_DOWN if direction is down

5.21.2.3 bool is_emergency_button_pressed (void)

Checks if the emeregency button is pressed

Returns

EMERGENCY_NOT_PRESSED if button is not pressed
EMERGENCY_PRESSED if button is pressed

5.21.2.4 motor_running_e is_motor_running (void)

Checks if the elevator is moving

Returns

MOTOR_RUNNING if elevator is moving
MOTOR_NOT_RUNNING if elevator is still

5.21.2.5 void set_elevator_light (int *floor*)

5.21.2.6 void set_motor_direction (motor_direction_e *dir*)

Set motor direction

Parameters

dir	is the desired direction of the motor

5.21.2.7 void start_motor (void)

Starts motor

5.21.2.8 void stop_motor (void)

Stops motor

5.21.2.9 void update_elevator_driver (bool *init_complete*)

Updates the module

Parameters

init_complete	Init is completed

5.21.3 Variable Documentation

5.21.3.1 motor_direction_e current_motor_direction = MOTOR_DIRECTION_UP

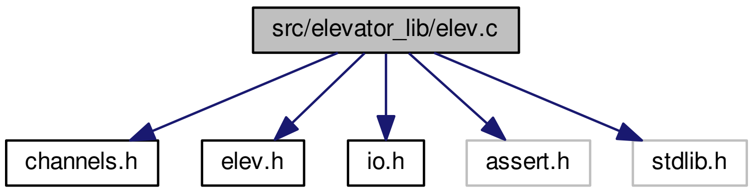
5.21.3.2 bool elevator_lights[NUM_FLOORS] = { false, false, false, false}

5.21.3.3 motor_running_e motor_running_status = MOTOR_NOT_RUNNING

5.22 src/elevator_lib/elev.c File Reference

```
#include "channels.h"
#include "elev.h"
#include "io.h"
#include <assert.h>
#include <stdlib.h>
```

Include dependency graph for elev.c:



Macros

- #define N_BUTTONS 3

Functions

- int elev_init (void)
- void elev_set_motor_direction (elev_motor_direction_t dirn)
- void elev_set_door_open_lamp (int value)
- int elev_get_obstruction_signal (void)
- int elev_get_stop_signal (void)
- void elev_set_stop_lamp (int value)
- int elev_get_floor_sensor_signal (void)
- void elev_set_floor_indicator (int floor)
- int elev_get_button_signal (elev_button_type_t button, int floor)
- void elev_set_button_lamp (elev_button_type_t button, int floor, int value)

5.22.1 Macro Definition Documentation

5.22.1.1 #define N_BUTTONS 3

5.22.2 Function Documentation

5.22.2.1 int elev_get_button_signal (elev_button_type_t *button,* int *floor*)

Gets a button signal.

Parameters

button	Which button type to check. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside the elevator).
floor	Which floor to check button. Must be 0-3.

Returns

0 if button is not pushed. 1 if button is pushed.

5.22.2.2 int elev_get_floor_sensor_signal (void)

Get floor sensor signal.

Returns

-1 if elevator is not on a floor. 0-3 if elevator is on floor. 0 is ground floor, 3 is top floor.

5.22.2.3 int elev_get_obstruction_signal (void)

Get signal from obstruction switch.

Returns

1 if obstruction is enabled. 0 if not.

5.22.2.4 int elev_get_stop_signal (void)

Get signal from stop button.

Returns

1 if stop button is pushed, 0 if not.

5.22.2.5 int elev_init (void)

Initialize elevator.

Returns

Non-zero on success, 0 on failure.

5.22.2.6 void elev_set_button_lamp (elev_button_type_t *button,* int *floor,* int *value*)

Set a button lamp.

Parameters

lamp	Which type of lamp to set. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside" the elevator).
floor	Floor of lamp to set. Must be 0-3
value	Non-zero value turns lamp on, 0 turns lamp off.

5.22.2.7 void elev_set_door_open_lamp (int *value*)

Turn door-open lamp on or off.

Parameters

value	Non-zero value turns lamp on, 0 turns lamp off.

5.22.2.8 void elev_set_floor_indicator (int *floor*)

Set floor indicator lamp for a given floor.

Parameters

floor	Which floor lamp to turn on. Other floor lamps are turned off.

5.22.2.9 void elev_set_motor_direction (elev_motor_direction_t *dirn*)

Sets the motor direction of the elevator.

Parameters

dirn	New direction of the elevator.

5.22.2.10 void elev_set_stop_lamp (int *value*)

Turn stop lamp on or off.

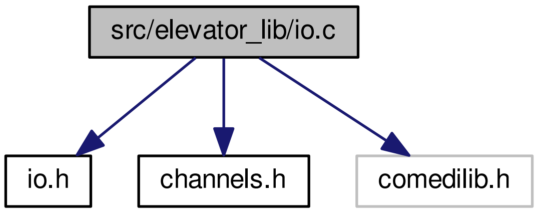
Parameters

value	Non-zero value turns lamp on, 0 turns lamp off.

5.23 src/elevator_lib/io.c File Reference

```
#include "io.h"
#include "channels.h"
#include <comedilib.h>
```

Include dependency graph for io.c:



Functions

- int io_init ()
- void io_set_bit (int channel)
- void io_clear_bit (int channel)
- void io_write_analog (int channel, int value)
- int io_read_bit (int channel)
- int io_read_analog (int channel)

5.23.1 Function Documentation

5.23.1.1 void io_clear_bit (int *channel*)

Clears a digital channel bit.

Parameters

channel	Channel bit to set.

5.23.1.2 int io_init ( )

Initialize libComedi in "Sanntidssalen"

Returns

Non-zero on success and 0 on failure

5.23.1.3 int io_read_analog (int *channel*)

Reads a bit value from an analog channel.

Parameters

channel	Channel to read from.

Returns

Value read.

5.23.1.4 int io_read_bit (int *channel*)

Reads a bit value from a digital channel.

Parameters

channel	Channel to read from.

Returns

Value read.

5.23.1.5 void io_set_bit (int *channel*)

Sets a digital channel bit.

Parameters

channel	Channel bit to set.

5.23.1.6 void io_write_analog (int *channel,* int *value*)

Writes a value to an analog channel.

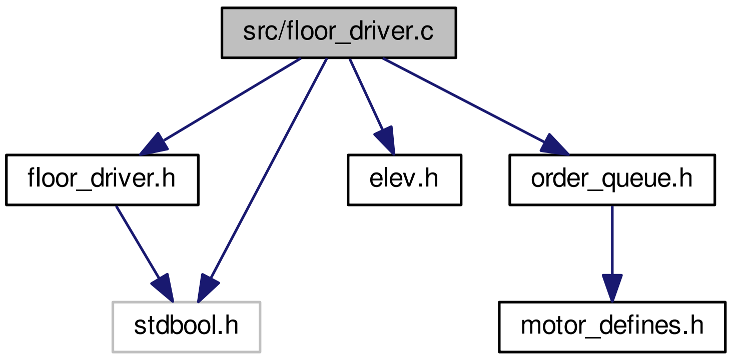
Parameters

channel	Channel to write to.
value	Value to write.

5.24 src/floor_driver.c File Reference

```
#include "floor_driver.h"
#include "elev.h"
#include "order_queue.h"
#include <stdbool.h>
```

Include dependency graph for floor_driver.c:



Functions

- void clear_floor_light (int floor)
- void update_floor_driver (bool init_complete)
- int get_current_floor (void)
- void set_floor_indicator (int floor)

Variables

- bool btn_light_state [2][4]

5.24.1 Function Documentation

5.24.1.1 void clear_floor_light (int *floor*)

Clear light at floor

Parameters

floor	Which floor light to clear

5.24.1.2 int get_current_floor (void)

Check if elevator is at a floor

Returns

current floor if at a floor, -1 if between floors

5.24.1.3 void set_floor_indicator (int *floor*)

Parameters

desired	floor

5.24.1.4 void update_floor_driver (bool *init_complete*)

Floor driver main function

Parameters

init_complete	Init is completed

5.24.2 Variable Documentation

5.24.2.1 bool btn_light_state[2][4]

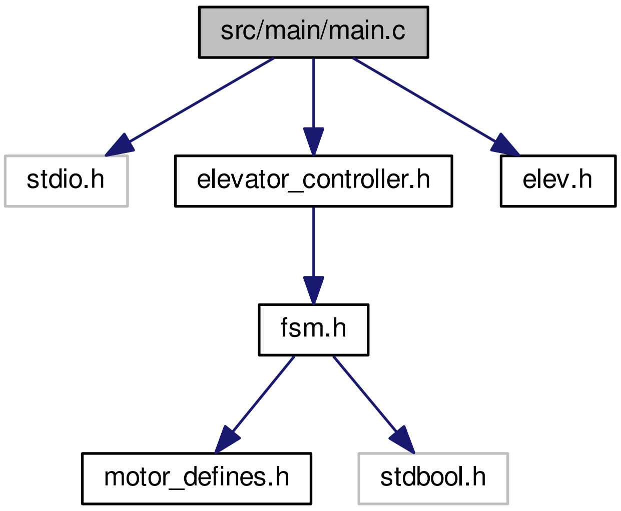
Initial value:

```
= {
    { false, false, false, false },
    { false, false, false, false }
}
```

5.25 src/main/main.c File Reference

```
#include <stdio.h>
#include "elevator_controller.h"
#include "elev.h"
```

Include dependency graph for main.c:



Functions

- int main ()

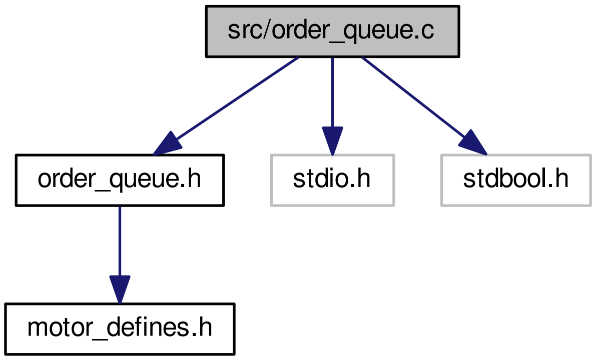
5.25.1 Function Documentation

5.25.1.1 int main ( )

5.26 src/order_queue.c File Reference

```
#include "order_queue.h"
#include <stdio.h>
#include <stdbool.h>
```

Include dependency graph for order_queue.c:



Macros

- #define NUM_FLOORS 4
- #define INVALID_VALUE -1

Enumerations

- enum has_order_e { NO_ORDER = 0, ORDER = 1 }

Functions

- bool valid_floor (int floor)
- void add_to_order_queue_up (int floor)
- void add_to_order_queue_down (int floor)
- void add_to_order_queue_dest (int floor)
- void empty_queue (void)

  Empty all queues.
- int get_next_order (int current_floor, motor_direction_e dir)
- void clear_order_in_queue (int floor)

Variables

- has_order_e orders_up [NUM_FLOORS] = { 0, 0, 0, 0}
- has_order_e orders_down [NUM_FLOORS] = { 0, 0, 0, 0}
- has_order_e orders_destination [NUM_FLOORS] = { 0, 0, 0, 0 }

5.26.1 Macro Definition Documentation

5.26.1.1 #define INVALID_VALUE -1

5.26.1.2 #define NUM_FLOORS 4

5.26.2 Enumeration Type Documentation

5.26.2.1 enum has_order_e

Enumerator

NO_ORDER

ORDER

5.26.3 Function Documentation

5.26.3.1 void add_to_order_queue_dest (int *floor*)

Add down order from floor

Parameters

floor	- Which floor ordered from

5.26.3.2 void add_to_order_queue_down (int *floor*)

Add down order from floor

Parameters

floor	- Which floor ordered from

5.26.3.3 void add_to_order_queue_up (int *floor*)

Add up order from floor

Parameters

floor	- Which floor ordered from

5.26.3.4 void clear_order_in_queue (int *floor*)

Clear all orders from given floor

Parameters

floor	Floor to clear

5.26.3.5 void empty_queue (void)

Empty all queues.

5.26.3.6 int get_next_order (int *current_floor,* motor_direction_e *dir*)

Get next order from queues

Parameters

current_floor	
dir	motor direction

Returns

next floor to stop, -1 if no orders in queue

5.26.3.7 bool valid_floor (int *floor*)

5.26.4 Variable Documentation

5.26.4.1 has_order_e orders_destination[NUM_FLOORS] = { 0, 0, 0, 0 }

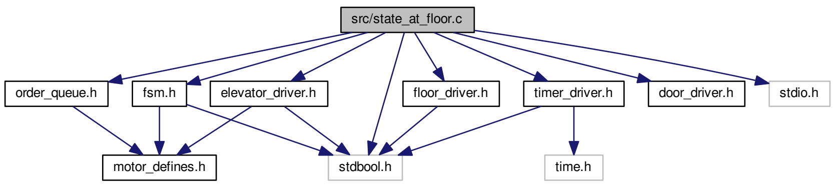
5.26.4.2 has_order_e orders_down[NUM_FLOORS] = { 0, 0, 0, 0}

5.26.4.3 has_order_e orders_up[NUM_FLOORS] = { 0, 0, 0, 0}

5.27 src/state_at_floor.c File Reference

```
#include "fsm.h"
#include "elevator_driver.h"
#include "floor_driver.h"
#include "door_driver.h"
#include "timer_driver.h"
#include <stdbool.h>
#include <stdio.h>
#include "order_queue.h"
```

Include dependency graph for state_at_floor.c:



Macros

- #define INVALID_VALUE -1

Functions

- fsm_state_e state_at_floor_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_at_floor_do (const state_data_t ∗state_data_p)

Variables

- timer_t current_timer

5.27.1 Macro Definition Documentation

5.27.1.1 #define INVALID_VALUE -1

5.27.2 Function Documentation

5.27.2.1 fsm_state_e state_at_floor_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←_p	Current system state

Returns

Next state

5.27.2.2 fsm_state_e state_at_floor_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

Returns

Next state

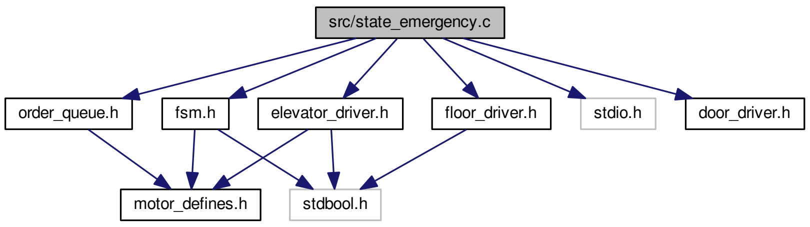
5.27.3 Variable Documentation

5.27.3.1 timer_t current_timer

5.28 src/state_emergency.c File Reference

```
#include <fsm.h>
#include "order_queue.h"
#include <stdio.h>
#include "floor_driver.h"
#include "elevator_driver.h"
#include "door_driver.h"
```

Include dependency graph for state_emergency.c:



Macros

- #define NUM_FLOORS 4
- #define INVALID_FLOOR -1

Functions

- fsm_state_e state_emergency_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_emergency_do (const state_data_t ∗state_data_p)

5.28.1 Macro Definition Documentation

5.28.1.1 #define INVALID_FLOOR -1

5.28.1.2 #define NUM_FLOORS 4

5.28.2 Function Documentation

5.28.2.1 fsm_state_e state_emergency_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data↵_p	Current system state

Returns

Next state

5.28.2.2 fsm_state_e state_emergency_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data↵_p	Current system state

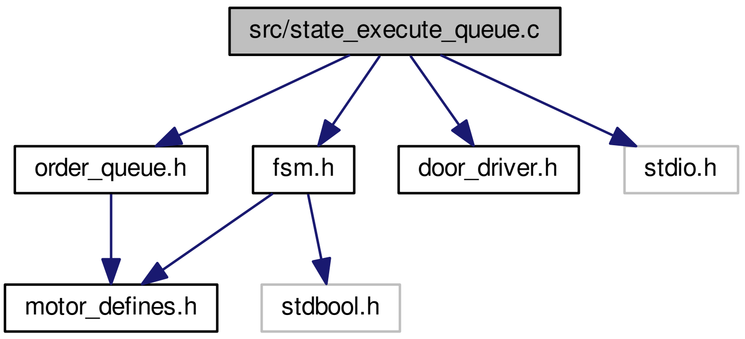
Returns

Next state

5.29 src/state_execute_queue.c File Reference

```
#include "fsm.h"
#include "order_queue.h"
#include "door_driver.h"
#include <stdio.h>
```

Include dependency graph for state_execute_queue.c:



Macros

- #define INVALID_VALUE -1

Functions

- fsm_state_e state_execute_queue_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_execute_queue_do (const state_data_t ∗state_data_p)

5.29.1 Macro Definition Documentation

5.29.1.1 #define INVALID_VALUE -1

5.29.2 Function Documentation

5.29.2.1 fsm_state_e state_execute_queue_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data↩ _p	Current system state

Returns

Next state

5.29.2.2 fsm_state_e state_execute_queue_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

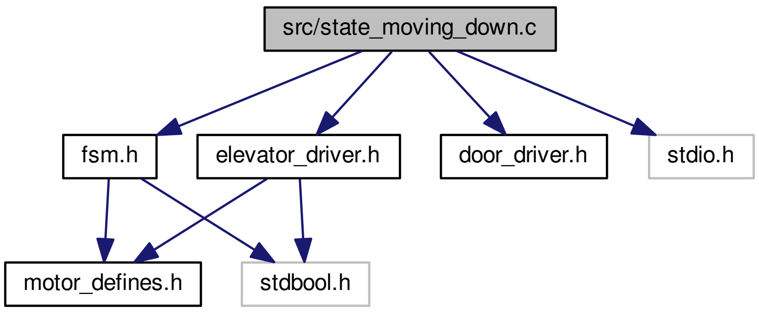
Returns

Next state

5.30 src/state_moving_down.c File Reference

```
#include <fsm.h>
#include "elevator_driver.h"
#include "door_driver.h"
#include <stdio.h>
```

Include dependency graph for state_moving_down.c:



Macros

- #define INVALID_FLOOR -1

Functions

- fsm_state_e state_moving_down_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_moving_down_do (const state_data_t ∗state_data_p)

5.30.1 Macro Definition Documentation

5.30.1.1 #define INVALID_FLOOR -1

5.30.2 Function Documentation

5.30.2.1 fsm_state_e state_moving_down_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←_p	Current system state

Returns

Next state

5.30.2.2 fsm_state_e state_moving_down_entry (const state_data_t ∗ *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

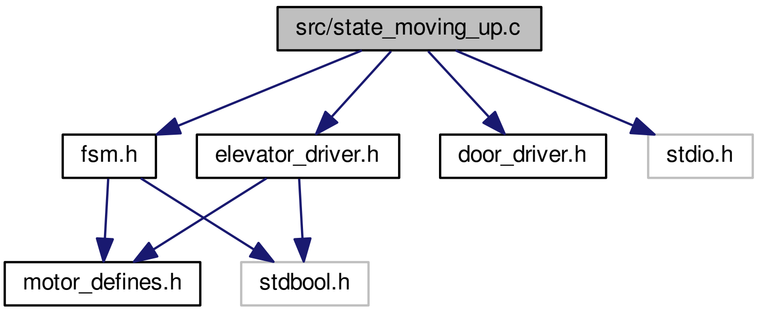
Returns

Next state

5.31 src/state_moving_up.c File Reference

```
#include <fsm.h>
#include "elevator_driver.h"
#include "door_driver.h"
#include <stdio.h>
```

Include dependency graph for state_moving_up.c:



Macros

- #define INVALID_FLOOR -1

Functions

- fsm_state_e state_moving_up_entry (const state_data_t ∗state_data_p)
- fsm_state_e state_moving_up_do (const state_data_t ∗state_data_p)

5.31.1 Macro Definition Documentation

5.31.1.1 #define INVALID_FLOOR -1

5.31.2 Function Documentation

5.31.2.1 fsm_state_e state_moving_up_do (const state_data_t ∗ *state_data_p*)

State do

Parameters

state_data←_p	Current system state

Returns

Next state

5.31.2.2 fsm_state_e state_moving_up_entry (const state_data_t * *state_data_p*)

State entry

Parameters

state_data←_p	Current system state

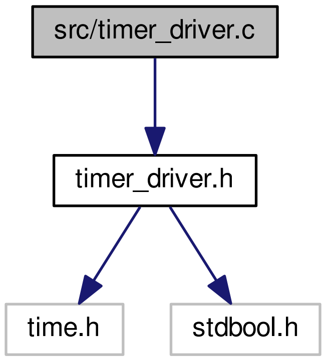
Returns

Next state

5.32 src/timer_driver.c File Reference

```
#include "timer_driver.h"
```

Include dependency graph for timer_driver.c:



Functions

- timer_t create_ms_timer (unsigned int d_ms)
- bool timer_has_elapsed (timer_t timer)

5.32.1 Function Documentation

5.32.1.1 timer_t create_ms_timer (unsigned int *d_ms*)

Create timer

Parameters

d_ms	Timer duration in milliseconds

Returns

Timer struct

5.32.1.2 bool timer_has_elapsed (timer_t *timer*)

Check if timer has elapsed

Parameters

timer	Timer to check

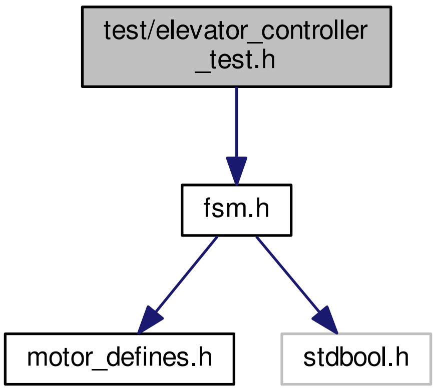
Returns

Has timer elapsed or not

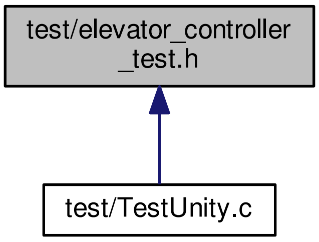
5.33 test/elevator_controller_test.h File Reference

```
#include "fsm.h"
```

Include dependency graph for elevator_controller_test.h:



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



Variables

- fsm_state_func state_table [FSM_NUM_STATES][FSM_NUM_STATES]

  Contains all state functions and transition functions.

5.33.1 Variable Documentation

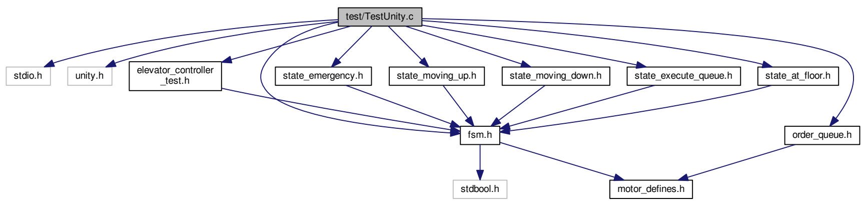
5.33.1.1 fsm_state_func state_table[FSM_NUM_STATES][FSM_NUM_STATES]

Contains all state functions and transition functions.

5.34 test/TestUnity.c File Reference

```
#include <stdio.h>
#include "unity.h"
#include "elevator_controller_test.h"
#include "fsm.h"
#include "order_queue.h"
#include "state_emergency.h"
#include "state_moving_up.h"
#include "state_moving_down.h"
#include "state_execute_queue.h"
#include "state_at_floor.h"
```

Include dependency graph for TestUnity.c:



Functions

- void test_sampletest (void)
- void test_to_state_at_floor (void)
- void test_to_moving_up_transitions (void)
- void test_to_moving_down_transitions (void)
- void test_to_emergency_transitions (void)
- void test_execute_queue_transitions (void)
- void test_order_queue (void)
- int main (int argc, char ∗∗argv)

5.34.1 Function Documentation

5.34.1.1 int main (int *argc,* char ∗∗ *argv*)

5.34.1.2 void test_execute_queue_transitions (void)

5.34.1.3 void test_order_queue (void)

5.34.1.4 void test_sampletest (void)

5.34.1.5 void test_to_emergency_transitions (void)

5.34.1.6 void test_to_moving_down_transitions (void)

5.34.1.7 void test_to_moving_up_transitions (void)

5.34.1.8 void test_to_state_at_floor (void)

Index