

Heisprosjekt TTK4235

Generated by Doxygen 1.8.11

Contents

1	HeisProsjekt	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	state_data_t Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Data Documentation	7
4.1.2.1	current_floor	7
4.1.2.2	emergency_button_pressed	8
4.1.2.3	last_floor	8
4.1.2.4	motor_direction	8
4.1.2.5	motor_running	8
4.1.2.6	target_floor	8
4.2	timer_t Struct Reference	8
4.2.1	Detailed Description	8
4.2.2	Member Data Documentation	8
4.2.2.1	duration_ms	8
4.2.2.2	start_time	8

5 File Documentation	9
5.1 CMakeLists.txt File Reference	9
5.2 include/channels.h File Reference	9
5.2.1 Macro Definition Documentation	10
5.2.1.1 BUTTON_COMMAND1	10
5.2.1.2 BUTTON_COMMAND2	10
5.2.1.3 BUTTON_COMMAND3	10
5.2.1.4 BUTTON_COMMAND4	10
5.2.1.5 BUTTON_DOWN1	10
5.2.1.6 BUTTON_DOWN2	10
5.2.1.7 BUTTON_DOWN3	10
5.2.1.8 BUTTON_DOWN4	10
5.2.1.9 BUTTON_UP1	10
5.2.1.10 BUTTON_UP2	11
5.2.1.11 BUTTON_UP3	11
5.2.1.12 BUTTON_UP4	11
5.2.1.13 LIGHT_COMMAND1	11
5.2.1.14 LIGHT_COMMAND2	11
5.2.1.15 LIGHT_COMMAND3	11
5.2.1.16 LIGHT_COMMAND4	11
5.2.1.17 LIGHT_DOOR_OPEN	11
5.2.1.18 LIGHT_DOWN1	11
5.2.1.19 LIGHT_DOWN2	11
5.2.1.20 LIGHT_DOWN3	11
5.2.1.21 LIGHT_DOWN4	11
5.2.1.22 LIGHT_FLOOR_IND1	11
5.2.1.23 LIGHT_FLOOR_IND2	11
5.2.1.24 LIGHT_STOP	11
5.2.1.25 LIGHT_UP1	11
5.2.1.26 LIGHT_UP2	11

5.2.1.27	LIGHT_UP3	11
5.2.1.28	LIGHT_UP4	11
5.2.1.29	MOTOR	11
5.2.1.30	MOTORDIR	11
5.2.1.31	OBSTRUCTION	11
5.2.1.32	PORT0	11
5.2.1.33	PORT1	12
5.2.1.34	PORT2	12
5.2.1.35	PORT3	12
5.2.1.36	PORT4	12
5.2.1.37	SENSOR_FLOOR1	12
5.2.1.38	SENSOR_FLOOR2	12
5.2.1.39	SENSOR_FLOOR3	12
5.2.1.40	SENSOR_FLOOR4	12
5.2.1.41	STOP	12
5.3	include/door_driver.h File Reference	12
5.3.1	Enumeration Type Documentation	12
5.3.1.1	door_state_e	12
5.3.2	Function Documentation	13
5.3.2.1	close_door(void)	13
5.3.2.2	is_door_open(void)	13
5.3.2.3	open_door(void)	13
5.4	include/elev.h File Reference	13
5.4.1	Macro Definition Documentation	14
5.4.1.1	N_FLOORS	14
5.4.2	Typedef Documentation	14
5.4.2.1	elev_button_type_t	14
5.4.2.2	elev_motor_direction_t	14
5.4.3	Enumeration Type Documentation	14
5.4.3.1	tag_elev_lamp_type	14

5.4.3.2	tag_elev_motor_direction	14
5.4.4	Function Documentation	14
5.4.4.1	elev_get_button_signal(elev_button_type_t button, int floor)	14
5.4.4.2	elev_get_floor_sensor_signal(void)	15
5.4.4.3	elev_get_obstruction_signal(void)	15
5.4.4.4	elev_get_stop_signal(void)	15
5.4.4.5	elev_init(void)	15
5.4.4.6	elev_set_button_lamp(elev_button_type_t button, int floor, int value)	15
5.4.4.7	elev_set_door_open_lamp(int value)	16
5.4.4.8	elev_set_floor_indicator(int floor)	16
5.4.4.9	elev_set_motor_direction(elev_motor_direction_t dirn)	16
5.4.4.10	elev_set_stop_lamp(int value)	16
5.5	include/elevator_controller.h File Reference	17
5.5.1	Function Documentation	17
5.5.1.1	elevator_controller_loop_once()	17
5.6	include/elevator_driver.h File Reference	18
5.6.1	Function Documentation	18
5.6.1.1	clear_elevator_light(int floor)	18
5.6.1.2	get_motor_direction(void)	19
5.6.1.3	is_emergency_button_pressed(void)	19
5.6.1.4	is_motor_running(void)	19
5.6.1.5	set_motor_direction(motor_direction_e dir)	19
5.6.1.6	start_motor(void)	19
5.6.1.7	stop_motor(void)	19
5.6.1.8	update_elevator_driver(bool init_complete)	20
5.7	include/floor_driver.h File Reference	21
5.7.1	Function Documentation	21
5.7.1.1	clear_floor_light(int floor)	21
5.7.1.2	get_current_floor(void)	22
5.7.1.3	set_floor_indicator(int floor)	22

5.7.1.4	<code>update_floor_driver(bool init_complete)</code>	22
5.8	<code>include/fsm.h</code> File Reference	22
5.8.1	Typedef Documentation	23
5.8.1.1	<code>fsm_state_func</code>	23
5.8.2	Enumeration Type Documentation	23
5.8.2.1	<code>fsm_state_e</code>	23
5.9	<code>include/io.h</code> File Reference	24
5.9.1	Function Documentation	24
5.9.1.1	<code>io_clear_bit(int channel)</code>	24
5.9.1.2	<code>io_init()</code>	24
5.9.1.3	<code>io_read_analog(int channel)</code>	24
5.9.1.4	<code>io_read_bit(int channel)</code>	25
5.9.1.5	<code>io_set_bit(int channel)</code>	25
5.9.1.6	<code>io_write_analog(int channel, int value)</code>	25
5.10	<code>include/motor_defines.h</code> File Reference	26
5.10.1	Enumeration Type Documentation	26
5.10.1.1	<code>motor_direction_e</code>	26
5.10.1.2	<code>motor_running_e</code>	26
5.11	<code>include/order_queue.h</code> File Reference	26
5.11.1	Function Documentation	27
5.11.1.1	<code>add_to_order_queue_dest(int floor)</code>	27
5.11.1.2	<code>add_to_order_queue_down(int floor)</code>	27
5.11.1.3	<code>add_to_order_queue_up(int floor)</code>	27
5.11.1.4	<code>clear_order_in_queue(int floor)</code>	28
5.11.1.5	<code>empty_queue(void)</code>	28
5.11.1.6	<code>get_next_order(int current_floor, motor_direction_e dir)</code>	28
5.12	<code>include/state_at_floor.h</code> File Reference	28
5.12.1	Function Documentation	29
5.12.1.1	<code>state_at_floor_do(const state_data_t *state_data_p)</code>	29
5.12.1.2	<code>state_at_floor_entry(const state_data_t *state_data_p)</code>	29

5.13	include/state_emergency.h File Reference	30
5.13.1	Function Documentation	30
5.13.1.1	state_emergency_do(const state_data_t *state_data_p)	30
5.13.1.2	state_emergency_entry(const state_data_t *state_data_p)	31
5.14	include/state_execute_queue.h File Reference	31
5.14.1	Function Documentation	32
5.14.1.1	state_execute_queue_do(const state_data_t *state_data_p)	32
5.14.1.2	state_execute_queue_entry(const state_data_t *state_data_p)	32
5.15	include/state_moving_down.h File Reference	33
5.15.1	Function Documentation	34
5.15.1.1	state_moving_down_do(const state_data_t *state_data_p)	34
5.15.1.2	state_moving_down_entry(const state_data_t *state_data_p)	34
5.16	include/state_moving_up.h File Reference	34
5.16.1	Function Documentation	35
5.16.1.1	state_moving_up_do(const state_data_t *state_data_p)	35
5.16.1.2	state_moving_up_entry(const state_data_t *state_data_p)	35
5.17	include/timer_driver.h File Reference	36
5.17.1	Function Documentation	37
5.17.1.1	create_ms_timer(unsigned int d_ms)	37
5.17.1.2	timer_has_elapsed(timer_t timer)	38
5.18	README.md File Reference	38
5.19	src/door_driver.c File Reference	38
5.19.1	Function Documentation	39
5.19.1.1	close_door(void)	39
5.19.1.2	is_door_open(void)	39
5.19.1.3	open_door(void)	39
5.19.2	Variable Documentation	39
5.19.2.1	door_state	39
5.20	src/elevator_controller.c File Reference	39
5.20.1	Function Documentation	40

5.20.1.1	elevator_controller_loop_once()	40
5.20.1.2	emergency_stop(void)	40
5.20.2	Variable Documentation	40
5.20.2.1	current_state	40
5.20.2.2	init_complete	40
5.20.2.3	last_floor	40
5.20.2.4	next_state	41
5.20.2.5	state_table	41
5.21	src/elevator_driver.c File Reference	41
5.21.1	Macro Definition Documentation	42
5.21.1.1	NUM_FLOORS	42
5.21.2	Function Documentation	42
5.21.2.1	clear_elevator_light(int floor)	42
5.21.2.2	get_motor_direction(void)	42
5.21.2.3	is_emergency_button_pressed(void)	43
5.21.2.4	is_motor_running(void)	43
5.21.2.5	set_elevator_light(int floor)	43
5.21.2.6	set_motor_direction(motor_direction_e dir)	43
5.21.2.7	start_motor(void)	43
5.21.2.8	stop_motor(void)	43
5.21.2.9	update_elevator_driver(bool init_complete)	43
5.21.3	Variable Documentation	44
5.21.3.1	current_motor_direction	44
5.21.3.2	elevator_lights	44
5.21.3.3	motor_running_status	44
5.22	src/elevator_lib/elev.c File Reference	44
5.22.1	Macro Definition Documentation	45
5.22.1.1	N_BUTTONS	45
5.22.2	Function Documentation	45
5.22.2.1	elev_get_button_signal(elev_button_type_t button, int floor)	45

5.22.2.2	elev_get_floor_sensor_signal(void)	46
5.22.2.3	elev_get_obstruction_signal(void)	46
5.22.2.4	elev_get_stop_signal(void)	46
5.22.2.5	elev_init(void)	46
5.22.2.6	elev_set_button_lamp(elev_button_type_t button, int floor, int value)	46
5.22.2.7	elev_set_door_open_lamp(int value)	47
5.22.2.8	elev_set_floor_indicator(int floor)	47
5.22.2.9	elev_set_motor_direction(elev_motor_direction_t dirn)	47
5.22.2.10	elev_set_stop_lamp(int value)	47
5.23	src/elevator_lib/io.c File Reference	48
5.23.1	Function Documentation	48
5.23.1.1	io_clear_bit(int channel)	48
5.23.1.2	io_init()	48
5.23.1.3	io_read_analog(int channel)	49
5.23.1.4	io_read_bit(int channel)	49
5.23.1.5	io_set_bit(int channel)	49
5.23.1.6	io_write_analog(int channel, int value)	49
5.24	src/floor_driver.c File Reference	50
5.24.1	Function Documentation	50
5.24.1.1	clear_floor_light(int floor)	50
5.24.1.2	get_current_floor(void)	51
5.24.1.3	set_floor_indicator(int floor)	51
5.24.1.4	update_floor_driver(bool init_complete)	51
5.24.2	Variable Documentation	51
5.24.2.1	btn_light_state	51
5.25	src/main/main.c File Reference	51
5.25.1	Function Documentation	52
5.25.1.1	main()	52
5.26	src/order_queue.c File Reference	52
5.26.1	Macro Definition Documentation	54

5.26.1.1	INVALID_VALUE	54
5.26.1.2	NUM_FLOORS	54
5.26.2	Enumeration Type Documentation	54
5.26.2.1	has_order_e	54
5.26.3	Function Documentation	54
5.26.3.1	add_to_order_queue_dest(int floor)	54
5.26.3.2	add_to_order_queue_down(int floor)	54
5.26.3.3	add_to_order_queue_up(int floor)	54
5.26.3.4	clear_order_in_queue(int floor)	54
5.26.3.5	empty_queue(void)	55
5.26.3.6	get_next_order(int current_floor, motor_direction_e dir)	55
5.26.3.7	valid_floor(int floor)	55
5.26.4	Variable Documentation	55
5.26.4.1	orders_destination	55
5.26.4.2	orders_down	55
5.26.4.3	orders_up	55
5.27	src/state_at_floor.c File Reference	55
5.27.1	Macro Definition Documentation	56
5.27.1.1	INVALID_VALUE	56
5.27.2	Function Documentation	56
5.27.2.1	state_at_floor_do(const state_data_t *state_data_p)	56
5.27.2.2	state_at_floor_entry(const state_data_t *state_data_p)	56
5.27.3	Variable Documentation	57
5.27.3.1	current_timer	57
5.28	src/state_emergency.c File Reference	57
5.28.1	Macro Definition Documentation	58
5.28.1.1	INVALID_FLOOR	58
5.28.1.2	NUM_FLOORS	58
5.28.2	Function Documentation	58
5.28.2.1	state_emergency_do(const state_data_t *state_data_p)	58

5.28.2.2	<code>state_emergency_entry(const state_data_t *state_data_p)</code>	58
5.29	<code>src/state_execute_queue.c</code> File Reference	58
5.29.1	Macro Definition Documentation	59
5.29.1.1	<code>INVALID_VALUE</code>	59
5.29.2	Function Documentation	59
5.29.2.1	<code>state_execute_queue_do(const state_data_t *state_data_p)</code>	59
5.29.2.2	<code>state_execute_queue_entry(const state_data_t *state_data_p)</code>	60
5.30	<code>src/state_moving_down.c</code> File Reference	60
5.30.1	Macro Definition Documentation	61
5.30.1.1	<code>INVALID_FLOOR</code>	61
5.30.2	Function Documentation	61
5.30.2.1	<code>state_moving_down_do(const state_data_t *state_data_p)</code>	61
5.30.2.2	<code>state_moving_down_entry(const state_data_t *state_data_p)</code>	61
5.31	<code>src/state_moving_up.c</code> File Reference	61
5.31.1	Macro Definition Documentation	62
5.31.1.1	<code>INVALID_FLOOR</code>	62
5.31.2	Function Documentation	62
5.31.2.1	<code>state_moving_up_do(const state_data_t *state_data_p)</code>	62
5.31.2.2	<code>state_moving_up_entry(const state_data_t *state_data_p)</code>	63
5.32	<code>src/timer_driver.c</code> File Reference	64
5.32.1	Function Documentation	64
5.32.1.1	<code>create_ms_timer(unsigned int d_ms)</code>	64
5.32.1.2	<code>timer_has_elapsed(timer_t timer)</code>	65
5.33	<code>test/elevator_controller_test.h</code> File Reference	65
5.33.1	Variable Documentation	66
5.33.1.1	<code>state_table</code>	66
5.34	<code>test/TestUnity.c</code> File Reference	66
5.34.1	Function Documentation	67
5.34.1.1	<code>main(int argc, char **argv)</code>	67
5.34.1.2	<code>test_execute_queue_transitions(void)</code>	67
5.34.1.3	<code>test_order_queue(void)</code>	67
5.34.1.4	<code>test_sampletest(void)</code>	67
5.34.1.5	<code>test_to_emergency_transitions(void)</code>	67
5.34.1.6	<code>test_to_moving_down_transitions(void)</code>	67
5.34.1.7	<code>test_to_moving_up_transitions(void)</code>	67
5.34.1.8	<code>test_to_state_at_floor(void)</code>	67

Chapter 1

HeisProsjekt

Heisprosjekt i C

How to build:

Using docker:

Install docker and run:

```
1 docker build .
```

to build and test code

Using CMake and make

Install CMake:

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

Create build directory and change directory

```
1 mkdir build
2 cd build
```

Run CMake with parent directory

```
1 cmake ..
```

Run make on generated makefile

```
1 make
```

Run executable

```
1 ./heisprosjekt
```

Using build script

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

```
1 ./build.sh
```

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

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

Run the built binary

```
1 ./heisprosjekt
```

How to build parent image

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

Chapter 2

Class Index

2.1 Class List

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

state_data_t	Struct to be passed in, containing useful data	7
timer_t	Timer struct	8

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

include/channels.h	9
include/door_driver.h	12
include/elev.h	13
include/elevator_controller.h	17
include/elevator_driver.h	18
include/floor_driver.h	21
include/fsm.h	22
include/io.h	24
include/motor_defines.h	26
include/order_queue.h	26
include/state_at_floor.h	28
include/state_emergency.h	30
include/state_execute_queue.h	31
include/state_moving_down.h	33
include/state_moving_up.h	34
include/timer_driver.h	36
src/door_driver.c	38
src/elevator_controller.c	39
src/elevator_driver.c	41
src/floor_driver.c	50
src/order_queue.c	52
src/state_at_floor.c	55
src/state_emergency.c	57
src/state_execute_queue.c	58
src/state_moving_down.c	60
src/state_moving_up.c	61
src/timer_driver.c	64
src/elevator_lib/elev.c	44
src/elevator_lib/io.c	48
src/main/main.c	51
test/elevator_controller_test.h	65
test/TestUnity.c	66

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](#)

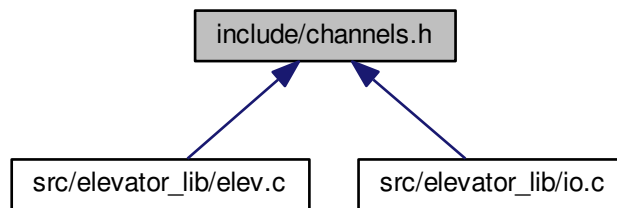
Chapter 5

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)`

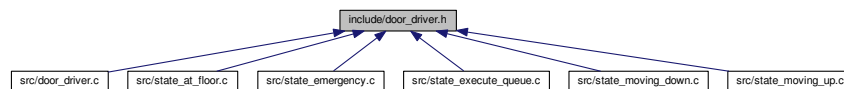
5.2.1.39 `#define SENSOR_FLOOR3 (0x200+6)`

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

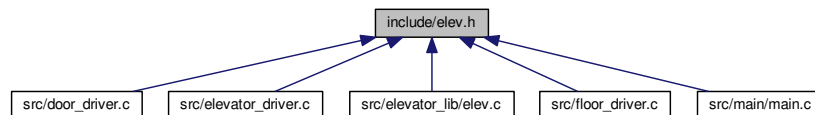
DOOR_OPEN if door is open
DOOR_CLOSED if door is closed

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

<i>button</i>	Which button type to check. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside the elevator).
<i>floor</i>	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

<i>lamp</i>	Which type of lamp to set. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside" the elevator).
<i>floor</i>	Floor of lamp to set. Must be 0-3
<i>value</i>	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

<i>value</i>	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

<i>floor</i>	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

<i>dirn</i>	New direction of the elevator.
-------------	--------------------------------

5.4.4.10 void elev_set_stop_lamp (int *value*)

Turn stop lamp on or off.

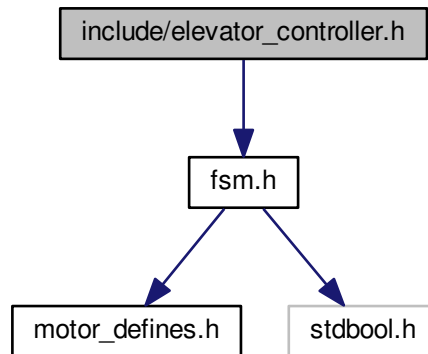
Parameters

<i>value</i>	Non-zero value turns lamp on, 0 turns lamp off.
--------------	---

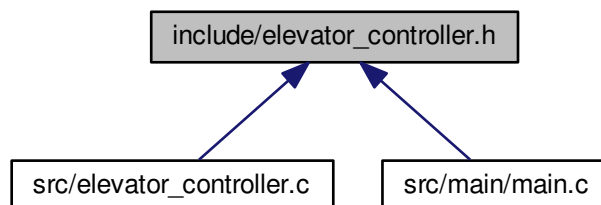
5.5 include/elevator_controller.h File Reference

```
#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

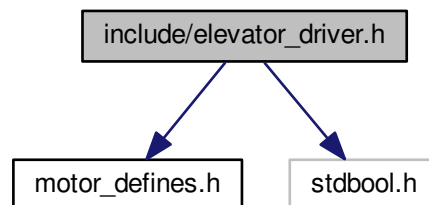
Run current state

5.6 include/elevator_driver.h File Reference

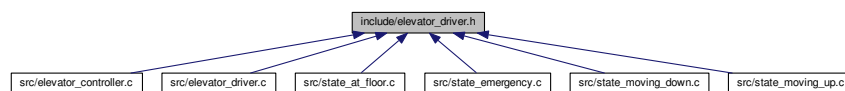
```
#include "motor_defines.h"
```

```
#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

<i>floor</i>	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 emergency 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

<i>dir</i>	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

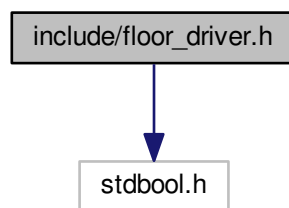
Parameters

<code>init_complete</code>	Init is completed
----------------------------	-------------------

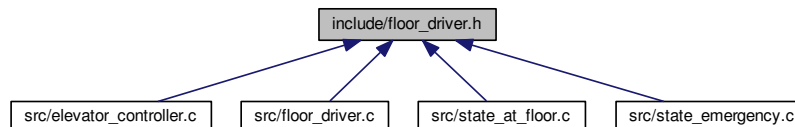
5.7 include/floor_driver.h File Reference

```
#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

<i>floor</i>	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**

<i>desired</i>	floor
----------------	-------

5.7.1.4 void update_floor_driver (bool *init_complete*)

Floor driver main function

Parameters

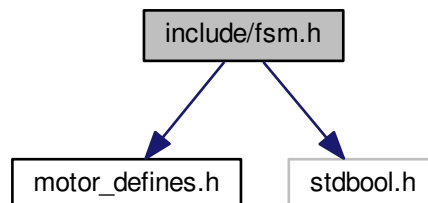
<i>init_complete</i>	Init is completed
----------------------	-------------------

5.8 include/fsm.h File Reference

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

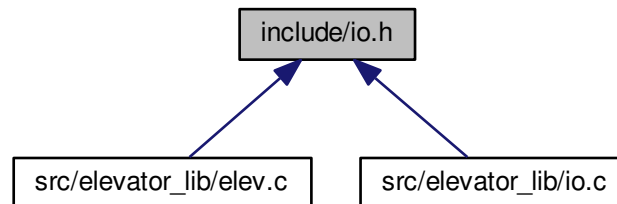
```
#include "stdbool.h"
```

Include dependency graph for fsm.h:



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

<i>channel</i>	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

<i>channel</i>	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

<i>channel</i>	Channel to read from.
----------------	-----------------------

Returns

Value read.

5.9.1.5 void io_set_bit (int *channel*)

Sets a digital channel bit.

Parameters

<i>channel</i>	Channel bit to set.
----------------	---------------------

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

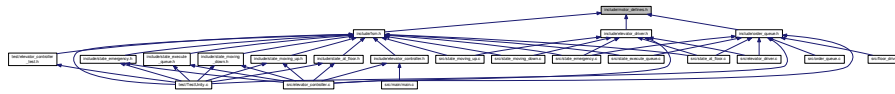
Writes a value to an analog channel.

Parameters

<i>channel</i>	Channel to write to.
<i>value</i>	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`

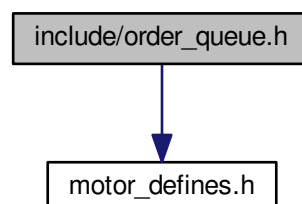
Enumerator

`MOTOR_NOT_RUNNING`
`MOTOR_RUNNING`

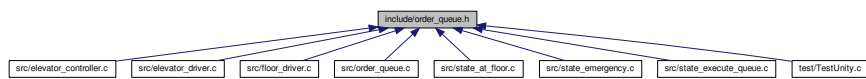
5.11 include/order_queue.h File Reference

```
#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

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.11.1.2 void [add_to_order_queue_down](#) (int *floor*)

Add down order from floor

Parameters

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.11.1.3 void [add_to_order_queue_up](#) (int *floor*)

Add up order from floor

Parameters

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.11.1.4 void clear_order_in_queue (int *floor*)

Clear all orders from given floor

Parameters

<i>floor</i>	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

<i>current_floor</i>	
<i>dir</i>	motor direction

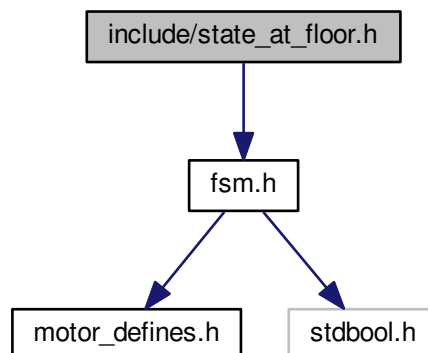
Returns

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

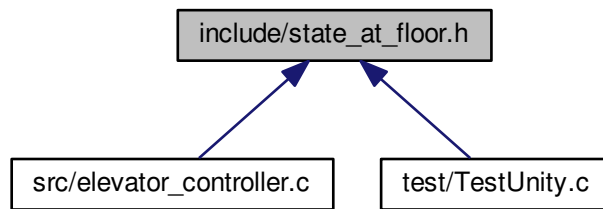
5.12 include/state_at_floor.h File Reference

```
#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

<i>state_data_t</i> * <i>state_data_p</i>	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

<i>state_data_t</i> * <i>state_data_p</i>	Current system state
---	----------------------

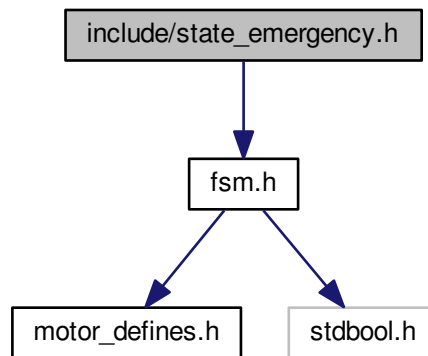
Returns

Next state

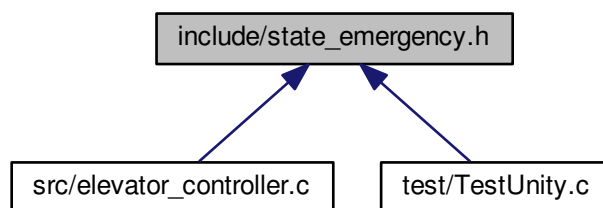
5.13 include/state_emergency.h File Reference

```
#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

<i>state_data</i> ↔ _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

<i>state_data</i> ↔ _p	Current system state
---------------------------	----------------------

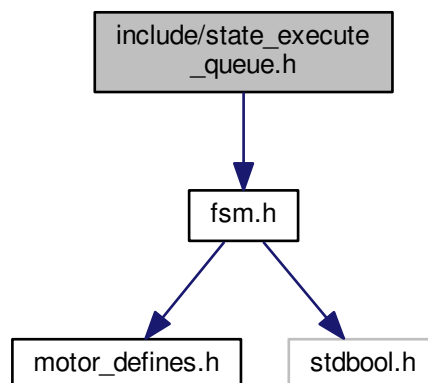
Returns

Next state

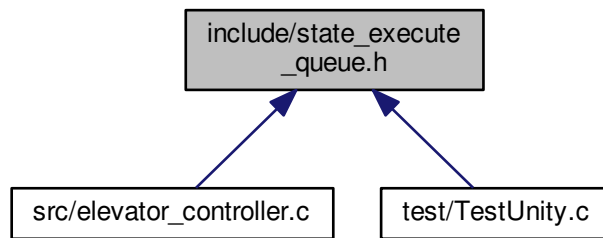
5.14 include/state_execute_queue.h File Reference

```
#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

<i>state_data_t</i> ↔ _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

<i>state_data_t</i> ↔ _p	Current system state
-----------------------------	----------------------

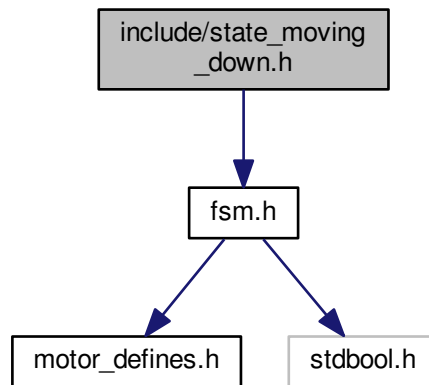
Returns

Next state

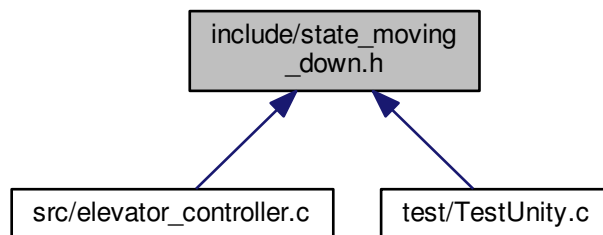
5.15 include/state_moving_down.h File Reference

```
#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

<i>state_data_p</i>	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

<i>state_data_p</i>	Current system state
---------------------	----------------------

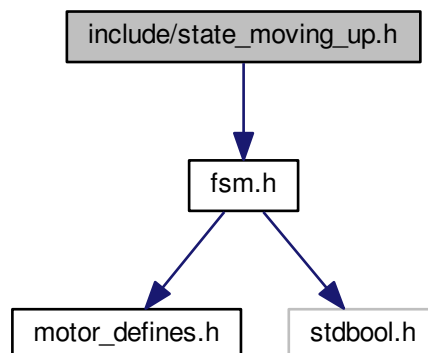
Returns

Next state

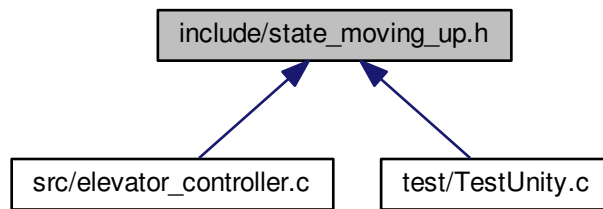
5.16 include/state_moving_up.h File Reference

```
#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

<i>state_data_p</i>	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

<i>state_data_p</i>	Current system state
---------------------	----------------------

Returns

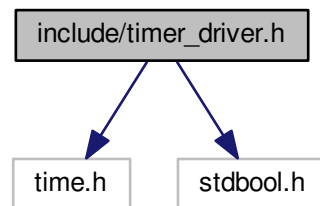
Next state

5.17 include/timer_driver.h File Reference

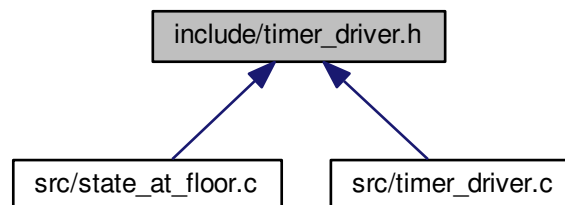
```
#include <time.h>
```

```
#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

<i>d_ms</i>	Timer duration in milliseconds
-------------	--------------------------------

Returns

Timer struct

5.17.1.2 bool timer_has_elapsed (timer_t timer)

Check if timer has elapsed

Parameters

<i>timer</i>	Timer to check
--------------	----------------

Returns

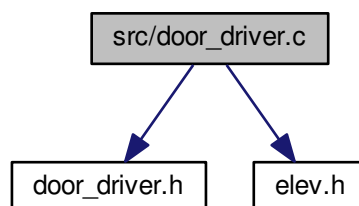
Has timer elapsed or not

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

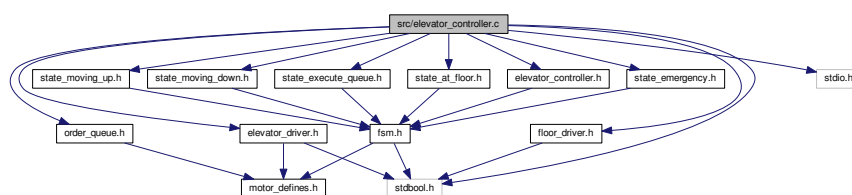
5.19.2 Variable Documentation

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]

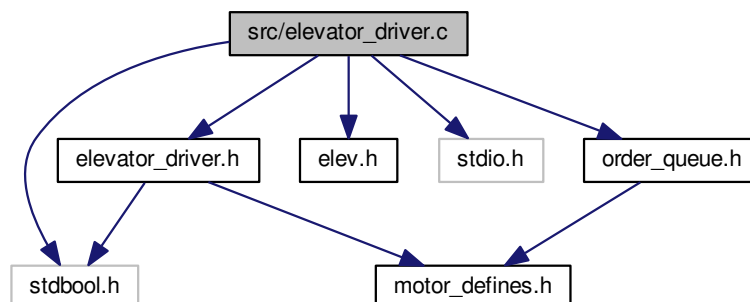
Initial value:

```
=
{
    { 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

<i>floor</i>	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 emergency 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

<i>dir</i>	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

<i>init_complete</i>	Init is completed
----------------------	-------------------

5.21.3 Variable Documentation

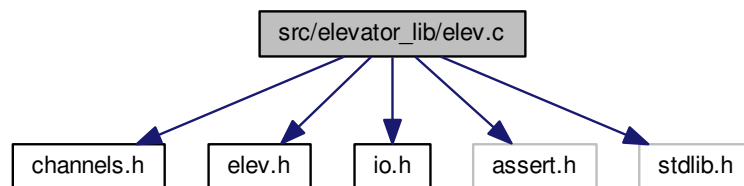
5.21.3.1 `motor_direction_e` `current_motor_direction = MOTOR_DIRECTION_UP`

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

<i>button</i>	Which button type to check. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside the elevator).
<i>floor</i>	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

<i>lamp</i>	Which type of lamp to set. Can be BUTTON_CALL_UP, BUTTON_CALL_DOWN or BUTTON_COMMAND (button "inside" the elevator).
<i>floor</i>	Floor of lamp to set. Must be 0-3
<i>value</i>	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

<i>value</i>	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

<i>floor</i>	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

<i>dirn</i>	New direction of the elevator.
-------------	--------------------------------

5.22.2.10 void elev_set_stop_lamp (int *value*)

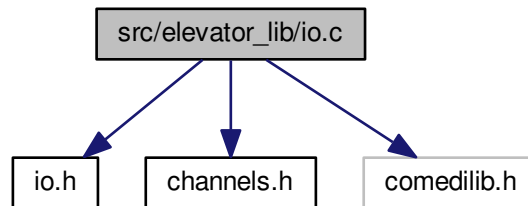
Turn stop lamp on or off.

Parameters

<i>value</i>	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

<i>channel</i>	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

<i>channel</i>	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

<i>channel</i>	Channel to read from.
----------------	-----------------------

Returns

Value read.

5.23.1.5 void io_set_bit (int *channel*)

Sets a digital channel bit.

Parameters

<i>channel</i>	Channel bit to set.
----------------	---------------------

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

Writes a value to an analog channel.

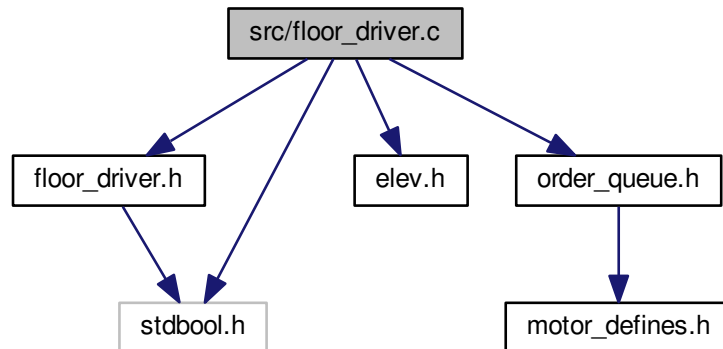
Parameters

<i>channel</i>	Channel to write to.
<i>value</i>	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

<i>floor</i>	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

<i>desired</i>	floor
----------------	-------

5.24.1.4 void update_floor_driver (bool init_complete)

Floor driver main function

Parameters

<i>init_complete</i>	Init is completed
----------------------	-------------------

5.24.2 Variable Documentation

5.24.2.1 bool btn_light_state[2][4]

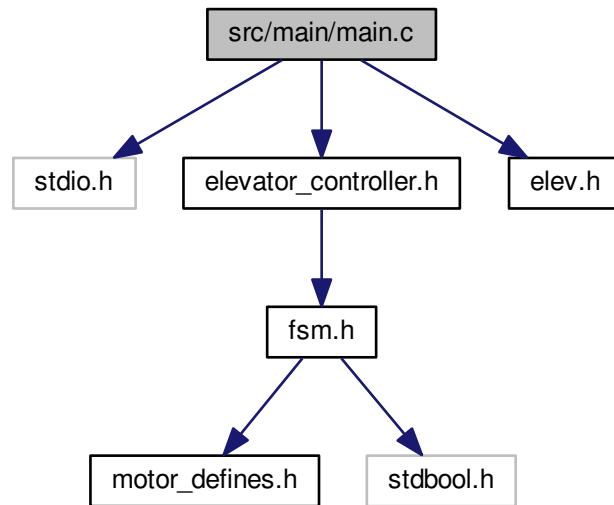
Initial value:

```
= {
    { 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` ()

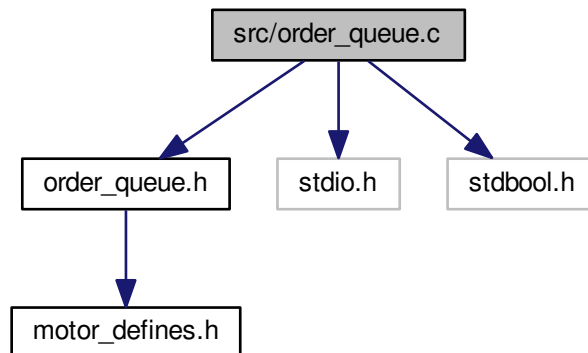
5.25.1 Function Documentation

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

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.26.3.2 `void add_to_order_queue_down (int floor)`

Add down order from floor

Parameters

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.26.3.3 `void add_to_order_queue_up (int floor)`

Add up order from floor

Parameters

<i>floor</i>	- Which floor ordered from
--------------	----------------------------

5.26.3.4 `void clear_order_in_queue (int floor)`

Clear all orders from given floor

Parameters

<i>floor</i>	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

<i>current_floor</i>	
<i>dir</i>	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 }

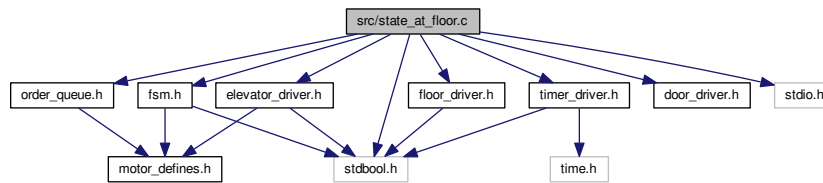
5.26.4.2 has_order_e orders_down[NUM_FLOORS] = { 0, 0, 0, 0 }

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

<code>state_data_p</code>	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

<code>state_data↔ _p</code>	Current system state
---------------------------------	----------------------

Returns

Next state

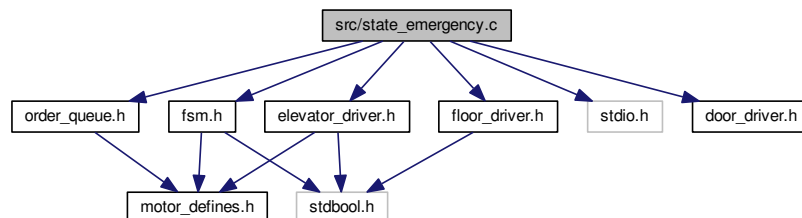
5.27.3 Variable Documentation

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

<code>state_data_↔ _p</code>	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

<code>state_data_↔ _p</code>	Current system state
----------------------------------	----------------------

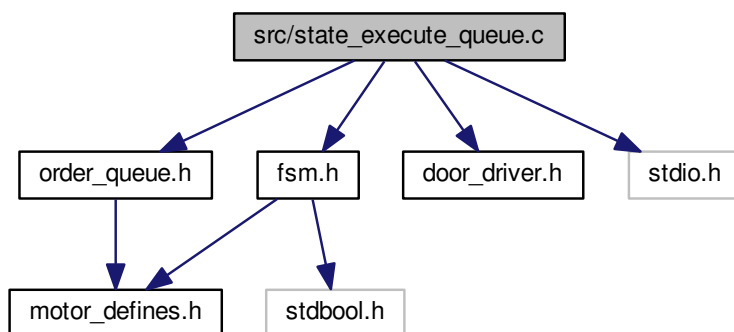
Returns

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

<code>state_data_p</code>	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

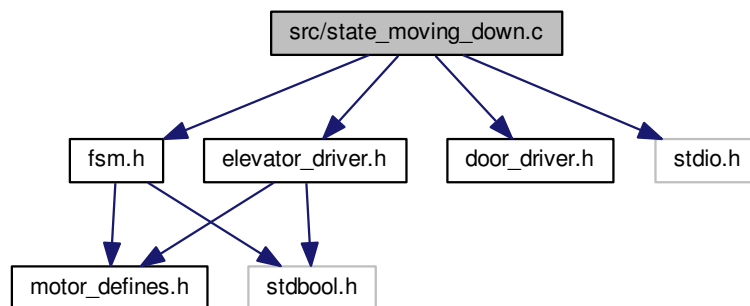
<i>state_data_p</i>	Current system state
---------------------	----------------------

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

<i>state_data_p</i>	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

<i>state_data_p</i>	Current system state
---------------------	----------------------

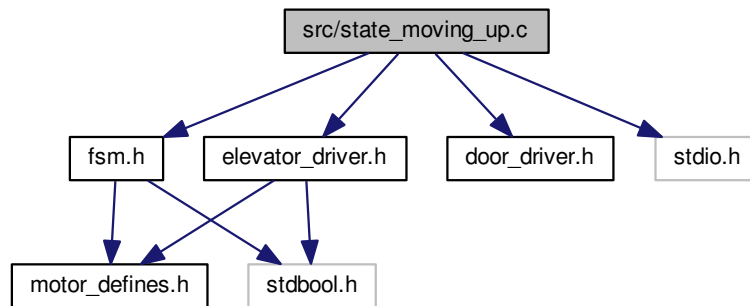
Returns

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

<code>state_data_p</code>	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

<i>state_data</i> _{<i>p</i>}	Current system state
---------------------------------------	----------------------

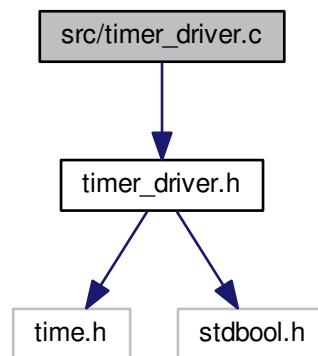
Returns

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

<i>d_ms</i>	Timer duration in milliseconds
-------------	--------------------------------

Returns

Timer struct

5.32.1.2 `bool timer_has_elapsed (timer_t timer)`

Check if timer has elapsed

Parameters

<i>timer</i>	Timer to check
--------------	----------------

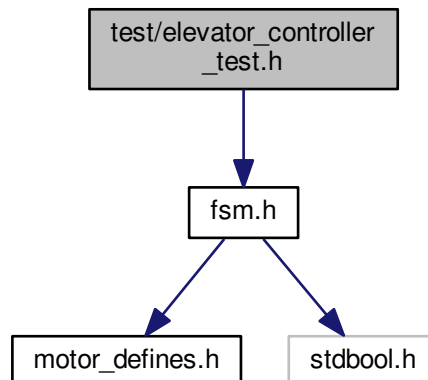
Returns

Has timer elapsed or not

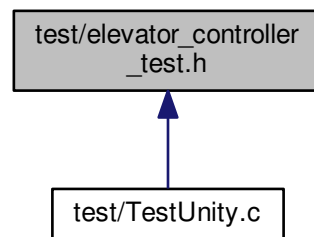
5.33 test/elevator_controller_test.h File Reference

```
#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.

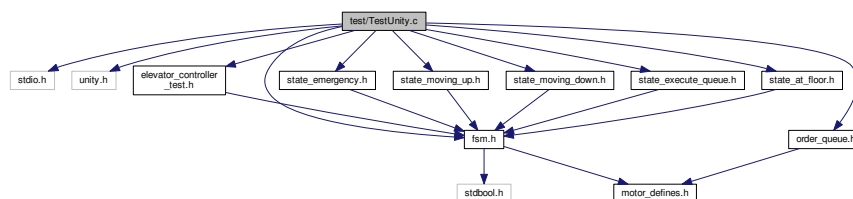
5.34 test/TestUnity.c File Reference

```

#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

add_to_order_queue_dest
 order_queue.c, [54](#)
 order_queue.h, [27](#)
add_to_order_queue_down
 order_queue.c, [54](#)
 order_queue.h, [27](#)
add_to_order_queue_up
 order_queue.c, [54](#)
 order_queue.h, [27](#)

BUTTON_CALL_DOWN
 elev.h, [14](#)
BUTTON_CALL_UP
 elev.h, [14](#)
BUTTON_COMMAND1
 channels.h, [10](#)
BUTTON_COMMAND2
 channels.h, [10](#)
BUTTON_COMMAND3
 channels.h, [10](#)
BUTTON_COMMAND4
 channels.h, [10](#)
BUTTON_COMMAND
 elev.h, [14](#)
BUTTON_DOWN1
 channels.h, [10](#)
BUTTON_DOWN2
 channels.h, [10](#)
BUTTON_DOWN3
 channels.h, [10](#)
BUTTON_DOWN4
 channels.h, [10](#)
BUTTON_UP1
 channels.h, [10](#)
BUTTON_UP2
 channels.h, [10](#)
BUTTON_UP3
 channels.h, [11](#)
BUTTON_UP4
 channels.h, [11](#)
btn_light_state
 floor_driver.c, [51](#)

CMakeLists.txt, [9](#)
channels.h
 BUTTON_COMMAND1, [10](#)
 BUTTON_COMMAND2, [10](#)
 BUTTON_COMMAND3, [10](#)
 BUTTON_COMMAND4, [10](#)
 BUTTON_DOWN1, [10](#)
 BUTTON_DOWN2, [10](#)
 BUTTON_DOWN3, [10](#)
 BUTTON_DOWN4, [10](#)
 BUTTON_UP1, [10](#)
 BUTTON_UP2, [10](#)
 BUTTON_UP3, [11](#)
 BUTTON_UP4, [11](#)
 LIGHT_COMMAND1, [11](#)
 LIGHT_COMMAND2, [11](#)
 LIGHT_COMMAND3, [11](#)
 LIGHT_COMMAND4, [11](#)
 LIGHT_DOOR_OPEN, [11](#)
 LIGHT_DOWN1, [11](#)
 LIGHT_DOWN2, [11](#)
 LIGHT_DOWN3, [11](#)
 LIGHT_DOWN4, [11](#)
 LIGHT_FLOOR_IND1, [11](#)
 LIGHT_FLOOR_IND2, [11](#)
 LIGHT_STOP, [11](#)
 LIGHT_UP1, [11](#)
 LIGHT_UP2, [11](#)
 LIGHT_UP3, [11](#)
 LIGHT_UP4, [11](#)
 MOTORDIR, [11](#)
 MOTOR, [11](#)
 OBSTRUCTION, [11](#)
 PORT0, [11](#)
 PORT1, [11](#)
 PORT2, [12](#)
 PORT3, [12](#)
 PORT4, [12](#)
 SENSOR_FLOOR1, [12](#)
 SENSOR_FLOOR2, [12](#)
 SENSOR_FLOOR3, [12](#)
 SENSOR_FLOOR4, [12](#)
 STOP, [12](#)
clear_elevator_light
 elevator_driver.c, [42](#)
 elevator_driver.h, [18](#)
clear_floor_light
 floor_driver.c, [50](#)
 floor_driver.h, [21](#)
clear_order_in_queue
 order_queue.c, [54](#)
 order_queue.h, [27](#)
close_door
 door_driver.c, [39](#)
 door_driver.h, [13](#)
create_ms_timer

- timer_driver.c, 64
- timer_driver.h, 37
- current_floor
 - state_data_t, 7
- current_motor_direction
 - elevator_driver.c, 44
- current_state
 - elevator_controller.c, 40
- current_timer
 - state_at_floor.c, 57
- DIRN_DOWN
 - elev.h, 14
- DIRN_STOP
 - elev.h, 14
- DIRN_UP
 - elev.h, 14
- DOOR_CLOSED
 - door_driver.h, 12
- DOOR_OPEN
 - door_driver.h, 12
- door_driver.c
 - close_door, 39
 - door_state, 39
 - is_door_open, 39
 - open_door, 39
- door_driver.h
 - close_door, 13
 - DOOR_CLOSED, 12
 - DOOR_OPEN, 12
 - door_state_e, 12
 - is_door_open, 13
 - open_door, 13
- door_state
 - door_driver.c, 39
- door_state_e
 - door_driver.h, 12
- duration_ms
 - timer_t, 8
- elev.c
 - elev_get_button_signal, 45
 - elev_get_floor_sensor_signal, 46
 - elev_get_obstruction_signal, 46
 - elev_get_stop_signal, 46
 - elev_init, 46
 - elev_set_button_lamp, 46
 - elev_set_door_open_lamp, 47
 - elev_set_floor_indicator, 47
 - elev_set_motor_direction, 47
 - elev_set_stop_lamp, 47
 - N_BUTTONS, 45
- elev.h
 - BUTTON_CALL_DOWN, 14
 - BUTTON_CALL_UP, 14
 - BUTTON_COMMAND, 14
 - DIRN_DOWN, 14
 - DIRN_STOP, 14
 - DIRN_UP, 14
 - elev_button_type_t, 14
 - elev_get_button_signal, 14
 - elev_get_floor_sensor_signal, 15
 - elev_get_obstruction_signal, 15
 - elev_get_stop_signal, 15
 - elev_init, 15
 - elev_motor_direction_t, 14
 - elev_set_button_lamp, 15
 - elev_set_door_open_lamp, 16
 - elev_set_floor_indicator, 16
 - elev_set_motor_direction, 16
 - elev_set_stop_lamp, 16
 - N_FLOORS, 14
 - tag_elev_lamp_type, 14
 - tag_elev_motor_direction, 14
- elev_button_type_t
 - elev.h, 14
- elev_get_button_signal
 - elev.c, 45
 - elev.h, 14
- elev_get_floor_sensor_signal
 - elev.c, 46
 - elev.h, 15
- elev_get_obstruction_signal
 - elev.c, 46
 - elev.h, 15
- elev_get_stop_signal
 - elev.c, 46
 - elev.h, 15
- elev_init
 - elev.c, 46
 - elev.h, 15
- elev_motor_direction_t
 - elev.h, 14
- elev_set_button_lamp
 - elev.c, 46
 - elev.h, 15
- elev_set_door_open_lamp
 - elev.c, 47
 - elev.h, 16
- elev_set_floor_indicator
 - elev.c, 47
 - elev.h, 16
- elev_set_motor_direction
 - elev.c, 47
 - elev.h, 16
- elev_set_stop_lamp
 - elev.c, 47
 - elev.h, 16
- elevator_controller.c
 - current_state, 40
 - elevator_controller_loop_once, 40
 - emergency_stop, 40
 - init_complete, 40
 - last_floor, 40
 - next_state, 40
 - state_table, 41
- elevator_controller.h

- elevator_controller_loop_once, 17
- elevator_controller_loop_once
 - elevator_controller.c, 40
 - elevator_controller.h, 17
- elevator_controller_test.h
 - state_table, 66
- elevator_driver.c
 - clear_elevator_light, 42
 - current_motor_direction, 44
 - elevator_lights, 44
 - get_motor_direction, 42
 - is_emergency_button_pressed, 42
 - is_motor_running, 43
 - motor_running_status, 44
 - NUM_FLOORS, 42
 - set_elevator_light, 43
 - set_motor_direction, 43
 - start_motor, 43
 - stop_motor, 43
 - update_elevator_driver, 43
- elevator_driver.h
 - clear_elevator_light, 18
 - get_motor_direction, 18
 - is_emergency_button_pressed, 19
 - is_motor_running, 19
 - set_motor_direction, 19
 - start_motor, 19
 - stop_motor, 19
 - update_elevator_driver, 19
- elevator_lights
 - elevator_driver.c, 44
- emergency_button_pressed
 - state_data_t, 7
- emergency_stop
 - elevator_controller.c, 40
- empty_queue
 - order_queue.c, 55
 - order_queue.h, 28
- FSM_NUM_STATES
 - fsm.h, 23
- floor_driver.c
 - btn_light_state, 51
 - clear_floor_light, 50
 - get_current_floor, 50
 - set_floor_indicator, 51
 - update_floor_driver, 51
- floor_driver.h
 - clear_floor_light, 21
 - get_current_floor, 22
 - set_floor_indicator, 22
 - update_floor_driver, 22
- fsm.h
 - FSM_NUM_STATES, 23
 - fsm_state_e, 23
 - fsm_state_func, 23
 - STATE_AT_FLOOR, 23
 - STATE_EMERGENCY, 23
 - STATE_EXECUTE_QUEUE, 23
 - STATE_MOVING_DOWN, 23
 - STATE_MOVING_UP, 23
 - fsm_state_e
 - fsm.h, 23
 - fsm_state_func
 - fsm.h, 23
 - get_current_floor
 - floor_driver.c, 50
 - floor_driver.h, 22
 - get_motor_direction
 - elevator_driver.c, 42
 - elevator_driver.h, 18
 - get_next_order
 - order_queue.c, 55
 - order_queue.h, 28
 - has_order_e
 - order_queue.c, 54
 - INVALID_FLOOR
 - state_emergency.c, 58
 - state_moving_down.c, 61
 - state_moving_up.c, 62
 - INVALID_VALUE
 - order_queue.c, 54
 - state_at_floor.c, 56
 - state_execute_queue.c, 59
 - include/channels.h, 9
 - include/door_driver.h, 12
 - include/elev.h, 13
 - include/elevator_controller.h, 17
 - include/elevator_driver.h, 18
 - include/floor_driver.h, 21
 - include/fsm.h, 22
 - include/io.h, 24
 - include/motor_defines.h, 26
 - include/order_queue.h, 26
 - include/state_at_floor.h, 28
 - include/state_emergency.h, 30
 - include/state_execute_queue.h, 31
 - include/state_moving_down.h, 33
 - include/state_moving_up.h, 34
 - include/timer_driver.h, 36
 - init_complete
 - elevator_controller.c, 40
 - io.c
 - io_clear_bit, 48
 - io_init, 48
 - io_read_analog, 48
 - io_read_bit, 49
 - io_set_bit, 49
 - io_write_analog, 49
 - io.h
 - io_clear_bit, 24
 - io_init, 24
 - io_read_analog, 24
 - io_read_bit, 25
 - io_set_bit, 25

- io_write_analog, [25](#)
- io_clear_bit
 - io.c, [48](#)
 - io.h, [24](#)
- io_init
 - io.c, [48](#)
 - io.h, [24](#)
- io_read_analog
 - io.c, [48](#)
 - io.h, [24](#)
- io_read_bit
 - io.c, [49](#)
 - io.h, [25](#)
- io_set_bit
 - io.c, [49](#)
 - io.h, [25](#)
- io_write_analog
 - io.c, [49](#)
 - io.h, [25](#)
- is_door_open
 - door_driver.c, [39](#)
 - door_driver.h, [13](#)
- is_emergency_button_pressed
 - elevator_driver.c, [42](#)
 - elevator_driver.h, [19](#)
- is_motor_running
 - elevator_driver.c, [43](#)
 - elevator_driver.h, [19](#)
- LIGHT_COMMAND1
 - channels.h, [11](#)
- LIGHT_COMMAND2
 - channels.h, [11](#)
- LIGHT_COMMAND3
 - channels.h, [11](#)
- LIGHT_COMMAND4
 - channels.h, [11](#)
- LIGHT_DOOR_OPEN
 - channels.h, [11](#)
- LIGHT_DOWN1
 - channels.h, [11](#)
- LIGHT_DOWN2
 - channels.h, [11](#)
- LIGHT_DOWN3
 - channels.h, [11](#)
- LIGHT_DOWN4
 - channels.h, [11](#)
- LIGHT_FLOOR_IND1
 - channels.h, [11](#)
- LIGHT_FLOOR_IND2
 - channels.h, [11](#)
- LIGHT_STOP
 - channels.h, [11](#)
- LIGHT_UP1
 - channels.h, [11](#)
- LIGHT_UP2
 - channels.h, [11](#)
- LIGHT_UP3
 - channels.h, [11](#)
- LIGHT_UP4
 - channels.h, [11](#)
- last_floor
 - elevator_controller.c, [40](#)
 - state_data_t, [8](#)
- MOTOR_DIRECTION_DOWN
 - motor_defines.h, [26](#)
- MOTOR_DIRECTION_UP
 - motor_defines.h, [26](#)
- MOTOR_NOT_RUNNING
 - motor_defines.h, [26](#)
- MOTOR_RUNNING
 - motor_defines.h, [26](#)
- MOTORDIR
 - channels.h, [11](#)
- MOTOR
 - channels.h, [11](#)
- main
 - main.c, [52](#)
 - TestUnity.c, [67](#)
- main.c
 - main, [52](#)
- motor_defines.h
 - MOTOR_DIRECTION_DOWN, [26](#)
 - MOTOR_DIRECTION_UP, [26](#)
 - MOTOR_NOT_RUNNING, [26](#)
 - MOTOR_RUNNING, [26](#)
 - motor_direction_e, [26](#)
 - motor_running_e, [26](#)
- motor_direction
 - state_data_t, [8](#)
- motor_direction_e
 - motor_defines.h, [26](#)
- motor_running
 - state_data_t, [8](#)
- motor_running_e
 - motor_defines.h, [26](#)
- motor_running_status
 - elevator_driver.c, [44](#)
- N_BUTTONS
 - elev.c, [45](#)
- N_FLOORS
 - elev.h, [14](#)
- NO_ORDER
 - order_queue.c, [54](#)
- NUM_FLOORS
 - elevator_driver.c, [42](#)
 - order_queue.c, [54](#)
 - state_emergency.c, [58](#)
- next_state
 - elevator_controller.c, [40](#)
- OBSTRUCTION
 - channels.h, [11](#)
- ORDER
 - order_queue.c, [54](#)
- open_door

- door_driver.c, 39
- door_driver.h, 13
- order_queue.c
 - add_to_order_queue_dest, 54
 - add_to_order_queue_down, 54
 - add_to_order_queue_up, 54
 - clear_order_in_queue, 54
 - empty_queue, 55
 - get_next_order, 55
 - has_order_e, 54
 - INVALID_VALUE, 54
 - NO_ORDER, 54
 - NUM_FLOORS, 54
 - ORDER, 54
 - orders_destination, 55
 - orders_down, 55
 - orders_up, 55
 - valid_floor, 55
- order_queue.h
 - add_to_order_queue_dest, 27
 - add_to_order_queue_down, 27
 - add_to_order_queue_up, 27
 - clear_order_in_queue, 27
 - empty_queue, 28
 - get_next_order, 28
- orders_destination
 - order_queue.c, 55
- orders_down
 - order_queue.c, 55
- orders_up
 - order_queue.c, 55
- PORT0
 - channels.h, 11
- PORT1
 - channels.h, 11
- PORT2
 - channels.h, 12
- PORT3
 - channels.h, 12
- PORT4
 - channels.h, 12
- README.md, 38
- SENSOR_FLOOR1
 - channels.h, 12
- SENSOR_FLOOR2
 - channels.h, 12
- SENSOR_FLOOR3
 - channels.h, 12
- SENSOR_FLOOR4
 - channels.h, 12
- STATE_AT_FLOOR
 - fsm.h, 23
- STATE_EMERGENCY
 - fsm.h, 23
- STATE_EXECUTE_QUEUE
 - fsm.h, 23
- STATE_MOVING_DOWN
 - fsm.h, 23
- STATE_MOVING_UP
 - fsm.h, 23
- STOP
 - channels.h, 12
- set_elevator_light
 - elevator_driver.c, 43
- set_floor_indicator
 - floor_driver.c, 51
 - floor_driver.h, 22
- set_motor_direction
 - elevator_driver.c, 43
 - elevator_driver.h, 19
- src/door_driver.c, 38
- src/elevator_controller.c, 39
- src/elevator_driver.c, 41
- src/elevator_lib/elev.c, 44
- src/elevator_lib/io.c, 48
- src/floor_driver.c, 50
- src/main/main.c, 51
- src/order_queue.c, 52
- src/state_at_floor.c, 55
- src/state_emergency.c, 57
- src/state_execute_queue.c, 58
- src/state_moving_down.c, 60
- src/state_moving_up.c, 61
- src/timer_driver.c, 64
- start_motor
 - elevator_driver.c, 43
 - elevator_driver.h, 19
- start_time
 - timer_t, 8
- state_at_floor.c
 - current_timer, 57
 - INVALID_VALUE, 56
 - state_at_floor_do, 56
 - state_at_floor_entry, 56
- state_at_floor.h
 - state_at_floor_do, 29
 - state_at_floor_entry, 29
- state_at_floor_do
 - state_at_floor.c, 56
 - state_at_floor.h, 29
- state_at_floor_entry
 - state_at_floor.c, 56
 - state_at_floor.h, 29
- state_data_t, 7
 - current_floor, 7
 - emergency_button_pressed, 7
 - last_floor, 8
 - motor_direction, 8
 - motor_running, 8
 - target_floor, 8
- state_emergency.c
 - INVALID_FLOOR, 58
 - NUM_FLOORS, 58
 - state_emergency_do, 58

- state_emergency_entry, 58
- state_emergency.h
 - state_emergency_do, 30
 - state_emergency_entry, 31
- state_emergency_do
 - state_emergency.c, 58
 - state_emergency.h, 30
- state_emergency_entry
 - state_emergency.c, 58
 - state_emergency.h, 31
- state_execute_queue.c
 - INVALID_VALUE, 59
 - state_execute_queue_do, 59
 - state_execute_queue_entry, 59
- state_execute_queue.h
 - state_execute_queue_do, 32
 - state_execute_queue_entry, 32
- state_execute_queue_do
 - state_execute_queue.c, 59
 - state_execute_queue.h, 32
- state_execute_queue_entry
 - state_execute_queue.c, 59
 - state_execute_queue.h, 32
- state_moving_down.c
 - INVALID_FLOOR, 61
 - state_moving_down_do, 61
 - state_moving_down_entry, 61
- state_moving_down.h
 - state_moving_down_do, 34
 - state_moving_down_entry, 34
- state_moving_down_do
 - state_moving_down.c, 61
 - state_moving_down.h, 34
- state_moving_down_entry
 - state_moving_down.c, 61
 - state_moving_down.h, 34
- state_moving_up.c
 - INVALID_FLOOR, 62
 - state_moving_up_do, 62
 - state_moving_up_entry, 62
- state_moving_up.h
 - state_moving_up_do, 35
 - state_moving_up_entry, 35
- state_moving_up_do
 - state_moving_up.c, 62
 - state_moving_up.h, 35
- state_moving_up_entry
 - state_moving_up.c, 62
 - state_moving_up.h, 35
- state_table
 - elevator_controller.c, 41
 - elevator_controller_test.h, 66
- stop_motor
 - elevator_driver.c, 43
 - elevator_driver.h, 19
- tag_elev_lamp_type
 - elev.h, 14
- tag_elev_motor_direction
 - elev.h, 14
- target_floor
 - state_data_t, 8
- test/TestUnity.c, 66
- test/elevator_controller_test.h, 65
- test_execute_queue_transitions
 - TestUnity.c, 67
- test_order_queue
 - TestUnity.c, 67
- test_sampletest
 - TestUnity.c, 67
- test_to_emergency_transitions
 - TestUnity.c, 67
- test_to_moving_down_transitions
 - TestUnity.c, 67
- test_to_moving_up_transitions
 - TestUnity.c, 67
- test_to_state_at_floor
 - TestUnity.c, 67
- TestUnity.c
 - main, 67
 - test_execute_queue_transitions, 67
 - test_order_queue, 67
 - test_sampletest, 67
 - test_to_emergency_transitions, 67
 - test_to_moving_down_transitions, 67
 - test_to_moving_up_transitions, 67
 - test_to_state_at_floor, 67
- timer_driver.c
 - create_ms_timer, 64
 - timer_has_elapsed, 65
- timer_driver.h
 - create_ms_timer, 37
 - timer_has_elapsed, 38
- timer_has_elapsed
 - timer_driver.c, 65
 - timer_driver.h, 38
- timer_t, 8
 - duration_ms, 8
 - start_time, 8
- update_elevator_driver
 - elevator_driver.c, 43
 - elevator_driver.h, 19
- update_floor_driver
 - floor_driver.c, 51
 - floor_driver.h, 22
- valid_floor
 - order_queue.c, 55