I2C Driver

Version 1.0.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 i2c_config_t Struct Reference

`#include <i2c_stm32f411_config.h>`

Data Fields

- i2c_enabled_t en
- i2c_ack_en_t ack_en
- uint32_t periph_clk_freq_MHz
- uint32_t i2c_op_freq_kHz
- i2c_fast_slow_t fast_or_std
- i2c_fm_duty_cycle_t duty_cycle

3.1.1 Detailed Description

Struct contains the settings required to configure an i2c device.

3.1.2 Field Documentation

3.1.2.1 ack_en

`i2c_ack_en_t ack_en`

Whether the device sends ACK upon byte reception

3.1.2.2 duty_cycle

`i2c_fm_duty_cycle_t duty_cycle`

The ratio of the period of low vs high cycles of bit pulses

3.1.2.3 en

```
i2c_enabled_t en
```

Whether the device is enabled or not

3.1.2.4 fast_or_std

```
i2c_fast_slow_t fast_or_std
```

Whether the I2C device will be in fast or standard mode

3.1.2.5 i2c_op_freq_kHz

```
uint32_t i2c_op_freq_kHz
```

The operational frequency of the I2C bus

3.1.2.6 periph_clk_freq_MHz

```
uint32_t periph_clk_freq_MHz
```

The frequency of the device in MHz

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

- /home/marko/Documents/embedded_workspace/i2c_driver/i2c_stm32f411_config.h

3.2 i2c_transfer_t Struct Reference

```
#include <i2c_interface.h>
```

Data Fields

- i2c_channel_t channel
- uint8_t * buffer
- uint32_t data_length
- uint8_t slave_address

3.2.1 Detailed Description

Generic transfer structure, independent of implementation. Passed into transmission functions.

3.2.2 Field Documentation

3.2.2.1 buffer

```
uint8_t* buffer
```

The data buffer

3.2.2.2 channel

```
i2c_channel_t channel
```

The target I2C peripheral

3.2.2.3 data_length

```
uint32_t data_length
```

The number of bytes to be receive/sent

3.2.2.4 slave_address

```
uint8_t slave_address
```

The 7-bit slave address

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

- /home/marko/Documents/embedded_workspace/i2c_driver/i2c_interface.h

Chapter 4

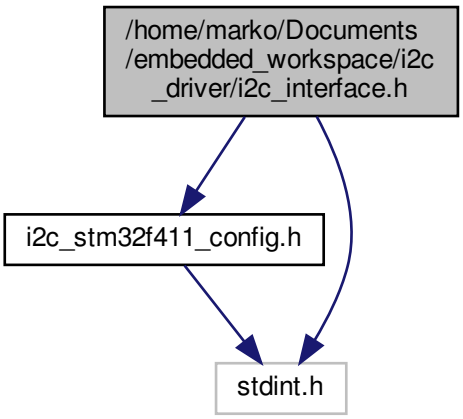
File Documentation

4.1 /home/marko/Documents/embedded_workspace/i2c_driver/i2c_interface.h File Reference

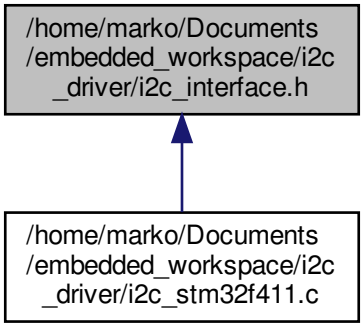
General interface covering user accesses to the i2c communication bus.

```
#include "i2c_stm32f411_config.h"
#include <stdint.h>
```

Include dependency graph for i2c_interface.h:



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



Data Structures

- struct i2c_transfer_t

Enumerations

- enum **i2c_interrupt_control_t** { **INTERRUPT_DISABLED**, **INTERRUPT_ENABLED** }

Functions

- void i2c_init (const i2c_config_t *config_table)
- void **i2c_irq_handler** (i2c_channel_t channel)
- void i2c_interrupt_control (i2c_channel_t channel, i2c_interrupt_dma_t interrupt, i2c_interrupt_control_t signal)
- void **i2c_master_transmit** (i2c_transfer_t *transfer)
- void **i2c_master_receive** (i2c_transfer_t *transfer)
- void **i2c_slave_transmit** (i2c_transfer_t *transfer)
- void **i2c_slave_receive** (i2c_transfer_t *transfer)
- void **i2c_master_transmit_it** (i2c_transfer_t *transfer)
- void **i2c_master_receive_it** (i2c_transfer_t *transfer)
- void **i2c_slave_transmit_it** (i2c_transfer_t *transfer)
- void **i2c_slave_receive_it** (i2c_transfer_t *transfer)
- void **i2c_register_write** (uint32_t i2c_register)
- uint32_t **i2c_register_read** (uint32_t i2c_regsister)

4.1.1 Detailed Description

General interface covering user accesses to the i2c communication bus.

4.1.2 Function Documentation

4.1.2.1 i2c_init()

```
void i2c_init (
            const i2c_config_t * config_table )
```

Description:

```
Carries out the initialisation of the I2C channels as per the information
in the config table
```

PRE-CONDITION: The config table has been obtained and is non-null

Returns

void

Example:

```
const i2c_config_t *config_table = i2c_config_get();
i2c_init(config_table);
```

See also

i2c_config_get
- CHANGE HISTORY -

Date	Software Version	Initials	Description

4.1.2.2 i2c_interrupt_control()

```
void i2c_interrupt_control (
            i2c_channel_t channel,
            i2c_interrupt_dma_t interrupt,
            i2c_interrupt_control_t signal )
```

Description:

```
Enabled or disables the selected interrupt on the selected channel. Caled both by users
and within the driver itself
```

PRE-CONDITION: The i2c_init function has been carried out successfully

Returns

void

Example:

```
i2c_interrupt_control(I2C_2, IT_BUF, INTERRUPT_ENABLED);
```

See also

i2c_init
i2c_master_transmit_it
i2c_master_receive_it
i2c_slave_transmit_it
i2c_slave_receive_it
- CHANGE HISTORY -

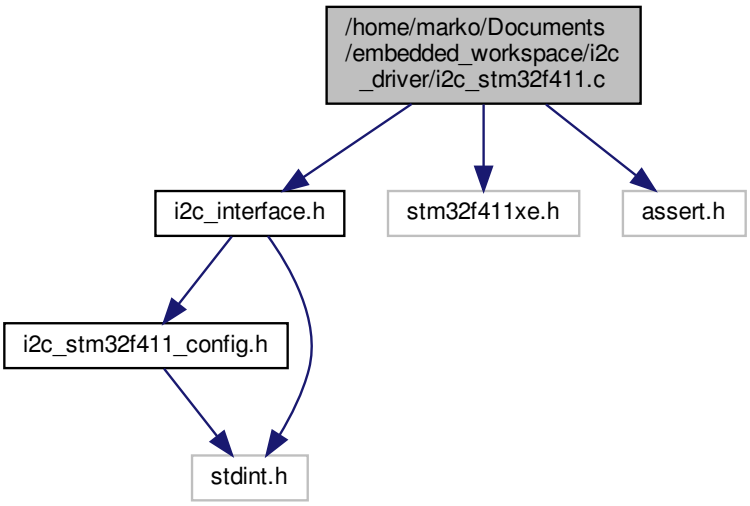
Date	Software Version	Initials	Description

4.2 /home/marko/Documents/embedded_workspace/i2c_driver/i2c_stm32f411.c File Reference

Chip specific implementation for i2c communication.

```
#include "i2c_interface.h"
#include "stm32f411xe.h"
#include <assert.h>
```

Include dependency graph for i2c_stm32f411.c:



Macros

- #define SM_RISE_TIME_MAX 1000
- #define FM_RISE_TIME_MAX 300

Typedefs

- typedef void(∗ i2c_interrupt_callback_t) (i2c_transfer_t ∗)

Functions

- void i2c_init (const i2c_config_t ∗config_table)
- void i2c_interrupt_control (i2c_channel_t channel, i2c_interrupt_dma_t interrupt, i2c_interrupt_control_t signal)
- void **i2c_master_transmit** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_master_receive** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_slave_transmit** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_slave_receive** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_master_transmit_it** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_master_receive_it** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_slave_transmit_it** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_slave_receive_it** (i2c_transfer_t ∗i2c_transfer)
- void **i2c_irq_handler** (i2c_channel_t channel)

4.2.1 Detailed Description

Chip specific implementation for i2c communication.

4.2.2 Macro Definition Documentation

4.2.2.1 FM_RISE_TIME_MAX

```
#define FM_RISE_TIME_MAX 300
```

Maximum rise time for a fast mode pulse in ns.

4.2.2.2 SM_RISE_TIME_MAX

```
#define SM_RISE_TIME_MAX 1000
```

Rise times obtained from the phillips i2c spec sheetMaximum rise time for a stanard mode pulse in ns.

4.2.3 Typedef Documentation

4.2.3.1 i2c_interrupt_callback_t

```
typedef void(* i2c_interrupt_callback_t) (i2c_transfer_t *)
```

Callback typedef for interrupt callbacks

4.2.4 Function Documentation

4.2.4.1 i2c_init()

```
void i2c_init (
            const i2c_config_t * config_table )
```

Description:

```
Carries out the initialisation of the I2C channels as per the information
in the config table
```

PRE-CONDITION: The config table has been obtained and is non-null

Returns

void

Example:

```
const i2c_config_t *config_table = i2c_config_get();
i2c_init(config_table);
```

See also

i2c_config_get
- CHANGE HISTORY -

Date	Software Version	Initials	Description

4.2.4.2 i2c_interrupt_control()

```
void i2c_interrupt_control (
            i2c_channel_t channel,
            i2c_interrupt_dma_t interrupt,
            i2c_interrupt_control_t signal )
```

Description:

```
Enabled or disables the selected interrupt on the selected channel. Caled both by users
and within the driver itself
```

PRE-CONDITION: The i2c_init function has been carried out successfully

Returns

void

Example:

```
i2c_interrupt_control(I2C_2, IT_BUF, INTERRUPT_ENABLED);
```

See also

i2c_init
i2c_master_transmit_it
i2c_master_receive_it
i2c_slave_transmit_it
i2c_slave_receive_it
- CHANGE HISTORY -

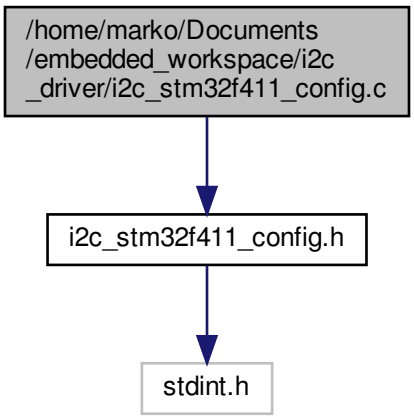
Date	Software Version	Initials	Description

4.3 /home/marko/Documents/embedded_workspace/i2c_driver/i2c_stm32f411_config.c File Reference

Contains the configuration information for each I2C channel.

```
#include "i2c_stm32f411_config.h"
```

Include dependency graph for i2c_stm32f411_config.c:



Functions

- const i2c_config_t * i2c_config_get (void)

4.3.1 Detailed Description

Contains the configuration information for each I2C channel.

4.3.2 Function Documentation

4.3.2.1 i2c_config_get()

```
const i2c_config_t* i2c_config_get (
            void  )
```

Description:

```
Returns a pointer to the base of the configuration table for i2c peripherals
```

PRE-CONDITION: The config table has been filled out and is non-null

Returns

∗i2c_config_t

Example:

```
const i2c_config_t *config_table = i2c_config_get();
i2c_init(config_table);
```

See also

i2c_init
- CHANGE HISTORY -

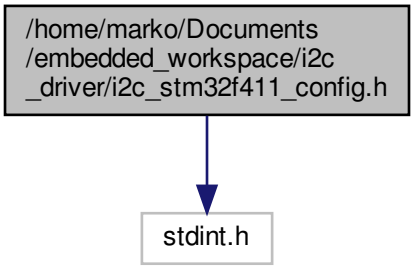
Date	Software Version	Initials	Description

4.4 /home/marko/Documents/embedded_workspace/i2c_driver/i2c_stm32f411_config.h File Reference

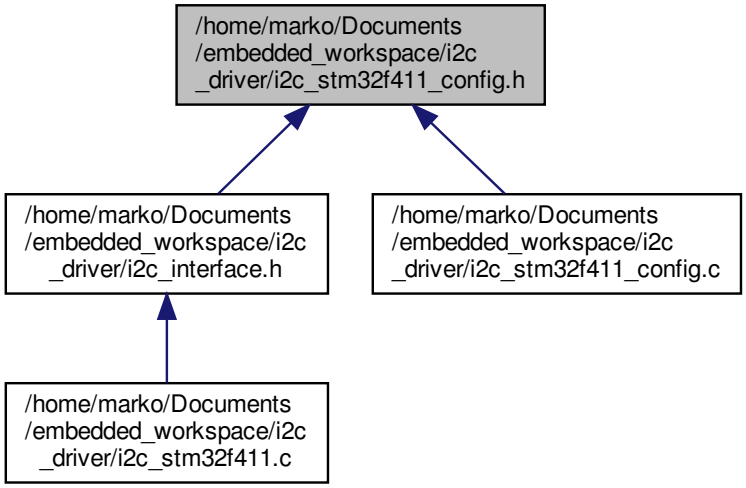
Contains the definitions and structures required to configure the i2c peripherals on an stm32f411.

```
#include <stdint.h>
```

Include dependency graph for i2c_stm32f411_config.h:



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



Data Structures

- struct i2c_config_t

Macros

- #define **DISABLED** 0
- #define **ENABLED** 1

Enumerations

- enum i2c_enabled_t { **I2C_DISABLED**, **I2C_ENABLED** }
- enum i2c_ack_en_t { **I2C_ACK_DISABLED**, **I2C_ACK_ENABLED** }
- enum i2c_interrupt_dma_t { IT_ERR, IT_EVT, IT_BUF, DMA_REQ }
- enum i2c_channel_t { **I2C_1** = 0x00UL, **I2C_2** = 0x01UL, **I2C_3** = 0x02UL, **NUM_I2C** }
- enum i2c_fast_slow_t { I2C_SM = 0x00UL, I2C_FM = 0x01UL }
- enum i2c_fm_duty_cycle_t { FM_MODE_2 = 0x00UL, FM_MODE_16_9 = 0x01UL }

Functions

- const i2c_config_t * i2c_config_get (void)

4.4.1 Detailed Description

Contains the definitions and structures required to configure the i2c peripherals on an stm32f411.

4.4.2 Enumeration Type Documentation

4.4.2.1 i2c_ack_en_t

`enum i2c_ack_en_t`

Options which decided whether the I2C returns an ACK pulse upon data reception or address match

4.4.2.2 i2c_channel_t

`enum i2c_channel_t`

Contains all of the I2C devices on chip

4.4.2.3 i2c_enabled_t

`enum i2c_enabled_t`

Options for enabling or disabling an I2C channel

4.4.2.4 i2c_fast_slow_t

`enum i2c_fast_slow_t`

Decides the maximum frequency with which the i2c may work

Enumerator

I2C_SM	Up to 100kHz
I2C_FM	Up to 400kHz

4.4.2.5 i2c_fm_duty_cycle_t

`enum i2c_fm_duty_cycle_t`

Determines the ratio of low to high periods per I2C pulse

Enumerator

FM_MODE_2	T_low/T_high = 2
FM_MODE_16_9	T_low/T_high = 16/9

4.4.2.6 i2c_interrupt_dma_t

```
enum i2c_interrupt_dma_t
```

Lists all the possible interrupts (and the dma request mode) available to the I2C channel

Enumerator

IT_ERR	The I2C raises an interrupt upon an error flag being raised
IT_EVT	The I2C raises an interrupt upon events: Start Bit, Address Matching, STOPF, BTF
IT_BUF	The I2C raises an interrupt when TxE or RxNE = 1 (if IT_EVT is also enabled)
DMA_REQ	Th2 I2C issues a DMA request upon TxE or TxNE = 1

4.4.3 Function Documentation

4.4.3.1 i2c_config_get()

```
const i2c_config_t* i2c_config_get (
            void  )
```

Description:

```
Returns a pointer to the base of the configuration table for i2c peripherals
```

PRE-CONDITION: The config table has been filled out and is non-null

Returns

∗i2c_config_t

Example:

```
const i2c_config_t *config_table = i2c_config_get();
i2c_init(config_table);
```

See also

i2c_init
- CHANGE HISTORY -

Date	Software Version	Initials	Description