Gyro SPI with STM32F427VIT6 - ARTPART Assessment

1.0

Generated by Doxygen 1.9.8

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

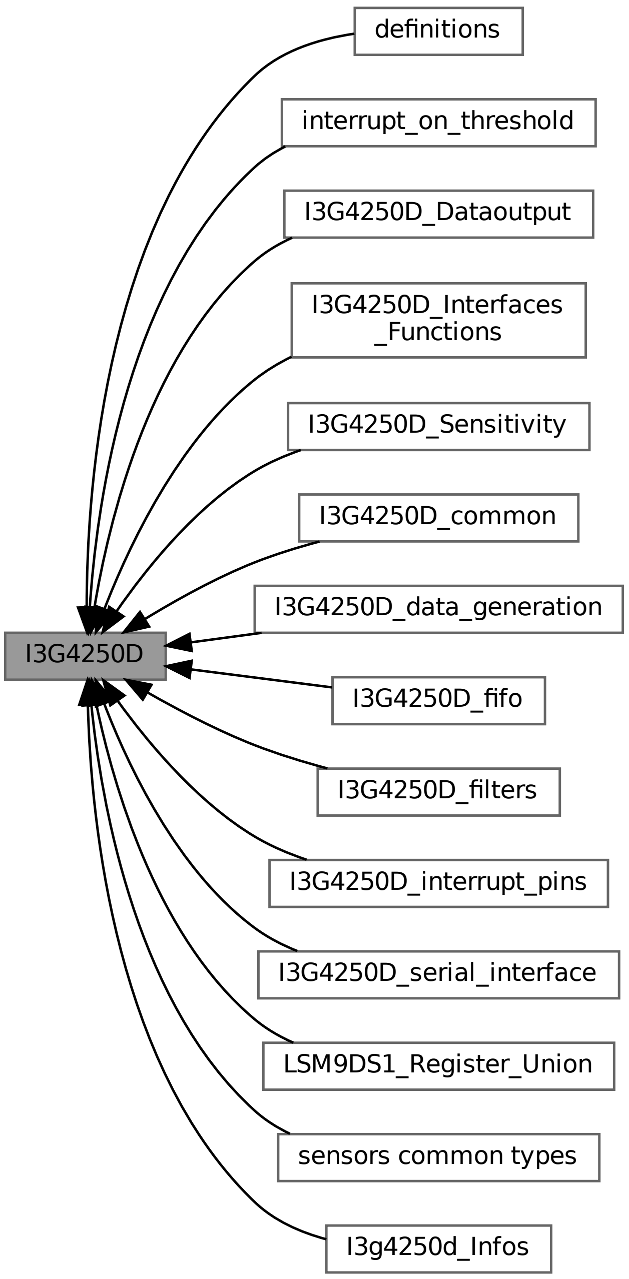
Chapter 4

Topic Documentation

4.1 I3G4250D

This file provides a set of functions needed to drive the i3g4250d enhanced inertial module.

Collaboration diagram for I3G4250D:



Modules

- I3G4250D_Interfaces_Functions

  This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.
- I3G4250D_Sensitivity

  These functions convert raw-data into engineering units.

- I3G4250D_data_generation

  This section groups all the functions concerning data generation.
- I3G4250D_Dataoutput

  This section groups all the data output functions.
- I3G4250D_common

  This section groups common useful functions.
- I3G4250D_filters

  This section group all the functions concerning the filters configuration.
- I3G4250D_serial_interface

  This section groups all the functions concerning main serial interface management.
- I3G4250D_interrupt_pins

  This section groups all the functions that manage interrupt pins.
- interrupt_on_threshold

  This section groups all the functions that manage the event generation on threshold.
- I3G4250D_fifo

  This section group all the functions concerning the fifo usage.
- definitions
- sensors common types
- I3g4250d_Infos
- LSM9DS1_Register_Union

  This union group all the registers having a bit-field description. This union is useful but it's not needed by the driver.

Data Structures

- struct i3g4250d_ctrl_reg1_t
- struct i3g4250d_ctrl_reg2_t
- struct i3g4250d_ctrl_reg3_t
- struct i3g4250d_ctrl_reg4_t
- struct i3g4250d_ctrl_reg5_t
- struct i3g4250d_fifo_ctrl_reg_t
- struct i3g4250d_fifo_src_reg_t
- struct i3g4250d_int1_cfg_t
- struct i3g4250d_int1_duration_t
- struct i3g4250d_int1_route_t
- struct i3g4250d_int1_src_t
- struct i3g4250d_int1_tsh_xh_t
- struct i3g4250d_int1_tsh_xl_t
- struct i3g4250d_int1_tsh_yh_t
- struct i3g4250d_int1_tsh_yl_t
- struct i3g4250d_int1_tsh_zh_t
- struct i3g4250d_int1_tsh_zl_t
- struct i3g4250d_int2_route_t
- struct i3g4250d_reference_t
- struct i3g4250d_status_reg_t

Macros

- #define __weak __attribute__((weak))
- #define I3G4250D_CTRL_REG1 0x20U
- #define I3G4250D_CTRL_REG2 0x21U
- #define I3G4250D_CTRL_REG3 0x22U
- #define I3G4250D_CTRL_REG4 0x23U
- #define I3G4250D_CTRL_REG5 0x24U
- #define I3G4250D_FIFO_CTRL_REG 0x2EU
- #define I3G4250D_FIFO_SRC_REG 0x2FU
- #define I3G4250D_INT1_CFG 0x30U
- #define I3G4250D_INT1_DURATION 0x38U
- #define I3G4250D_INT1_SRC 0x31U
- #define I3G4250D_INT1_TSH_XH 0x32U
- #define I3G4250D_INT1_TSH_XL 0x33U
- #define I3G4250D_INT1_TSH_YH 0x34U
- #define I3G4250D_INT1_TSH_YL 0x35U
- #define I3G4250D_INT1_TSH_ZH 0x36U
- #define I3G4250D_INT1_TSH_ZL 0x37U
- #define I3G4250D_OUT_TEMP 0x26U
- #define I3G4250D_OUT_X_H 0x29U
- #define I3G4250D_OUT_X_L 0x28U
- #define I3G4250D_OUT_Y_H 0x2BU
- #define I3G4250D_OUT_Y_L 0x2AU
- #define I3G4250D_OUT_Z_H 0x2DU
- #define I3G4250D_OUT_Z_L 0x2CU
- #define I3G4250D_REFERENCE 0x25U
- #define I3G4250D_STATUS_REG 0x27U
- #define I3G4250D_WHO_AM_I 0x0FU

Enumerations

- enum i3g4250d_and_or_t { I3G4250D_INT1_ON_TH_AND = 1 , I3G4250D_INT1_ON_TH_OR = 0 }
- enum i3g4250d_ble_t { I3G4250D_AUX_LSB_AT_LOW_ADD = 0 , I3G4250D_AUX_MSB_AT_LOW_ADD = 1 }
- enum i3g4250d_bw_t { I3G4250D_CUT_OFF_LOW = 0 , I3G4250D_CUT_OFF_MEDIUM = 1 , I3G4250D_CUT_OFF_HIGH = 2 , I3G4250D_CUT_OFF_VERY_HIGH = 3 }
- enum i3g4250d_dr_t {
  I3G4250D_ODR_OFF = 0x00 , I3G4250D_ODR_SLEEP = 0x08 , I3G4250D_ODR_100Hz = 0x0F , I3G4250D_ODR_200Hz = 0x1F ,
  I3G4250D_ODR_400Hz = 0x2F , I3G4250D_ODR_800Hz = 0x3F }
- enum i3g4250d_fifo_mode_t { I3G4250D_FIFO_BYPASS_MODE = 0x00 , I3G4250D_FIFO_MODE = 0x01 , I3G4250D_FIFO_STREAM_MODE = 0x02 }
- enum i3g4250d_fs_t { I3G4250D_245dps = 0x00 , I3G4250D_500dps = 0x01 , I3G4250D_2000dps = 0x02 }
- enum i3g4250d_h_lactive_t { I3G4250D_ACTIVE_HIGH = 0 , I3G4250D_ACTIVE_LOW = 1 }
- enum i3g4250d_hpcf_t {
  I3G4250D_HP_LEVEL_0 = 0 , I3G4250D_HP_LEVEL_1 = 1 , I3G4250D_HP_LEVEL_2 = 2 , I3G4250D_HP_LEVEL_3 = 3 ,
  I3G4250D_HP_LEVEL_4 = 4 , I3G4250D_HP_LEVEL_5 = 5 , I3G4250D_HP_LEVEL_6 = 6 , I3G4250D_HP_LEVEL_7 = 7 ,
  I3G4250D_HP_LEVEL_8 = 8 , I3G4250D_HP_LEVEL_9 = 9 }
- enum i3g4250d_hpm_t { I3G4250D_HP_NORMAL_MODE_WITH_RST = 0 , I3G4250D_HP_REFERENCE_SIGNAL = 1 , I3G4250D_HP_NORMAL_MODE = 2 , I3G4250D_HP_AUTO_RESET_ON_INT = 3 }
- enum i3g4250d_int1_sel_t { I3G4250D_ONLY_LPF1_ON_INT = 0 , I3G4250D_LPF1_HP_ON_INT = 1 , I3G4250D_LPF1_LPF2_ON_INT = 2 , I3G4250D_LPF1_HP_LPF2_ON_INT = 6 }

- enum i3g4250d_lir_t { I3G4250D_INT_PULSED = 0 , I3G4250D_INT_LATCHED = 1 }
- enum i3g4250d_out_sel_t { I3G4250D_ONLY_LPF1_ON_OUT = 0 , I3G4250D_LPF1_HP_ON_OUT = 1 , I3G4250D_LPF1_LPF2_ON_OUT = 2 , I3G4250D_LPF1_HP_LPF2_ON_OUT = 6 }
- enum i3g4250d_pp_od_t { I3G4250D_PUSH_PULL = 0 , I3G4250D_OPEN_DRAIN = 1 }
- enum i3g4250d_sim_t { I3G4250D_SPI_4_WIRE = 0 , I3G4250D_SPI_3_WIRE = 1 }
- enum i3g4250d_st_t { I3G4250D_GY_ST_DISABLE = 0 , I3G4250D_GY_ST_POSITIVE = 1 , I3G4250D_GY_ST_NEGATIVE = 3 }

Functions

- int32_t i3g4250d_angular_rate_raw_get (const stmdev_ctx_t ∗ctx, int16_t ∗val)

  Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].
- int32_t i3g4250d_axis_x_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_x_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_axis_y_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_y_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_axis_z_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_z_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_boot_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reboot memory content. Reload the calibration parameters.[get].
- int32_t i3g4250d_boot_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reboot memory content. Reload the calibration parameters.[set].
- int32_t i3g4250d_data_format_get (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t ∗val)

  Big/Little Endian data selection.[get].
- int32_t i3g4250d_data_format_set (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t val)

  Big/Little Endian data selection.[set].
- int32_t i3g4250d_data_rate_get (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t ∗val)

  Accelerometer data rate selection.[get].
- int32_t i3g4250d_data_rate_set (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t val)

  Accelerometer data rate selection.[set].
- int32_t i3g4250d_device_id_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Device Who amI.[get].
- int32_t i3g4250d_fifo_data_level_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO stored data level[get].
- int32_t i3g4250d_fifo_empty_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOemptybit.[get].
- int32_t i3g4250d_fifo_enable_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOenable.[get].
- int32_t i3g4250d_fifo_enable_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFOenable.[set].
- int32_t i3g4250d_fifo_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t ∗val)

  FIFO mode selection.[get].
- int32_t i3g4250d_fifo_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t val)

  FIFO mode selection.[set].
- int32_t i3g4250d_fifo_ovr_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Overrun bit status.[get].
- int32_t i3g4250d_fifo_watermark_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO watermark level selection.[get].
- int32_t i3g4250d_fifo_watermark_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFO watermark level selection.[set].
- int32_t i3g4250d_fifo_wtm_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)
- int32_t i3g4250d_filter_path_get (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t ∗val)

  Out/FIFO selection path. [get].
- int32_t i3g4250d_filter_path_internal_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t ∗val)

  Interrupt generator selection path.[get].
- int32_t i3g4250d_filter_path_internal_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t val)

  Interrupt generator selection path.[set].
- int32_t i3g4250d_filter_path_set (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t val)

  Out/FIFO selection path. [set].
- int32_t i3g4250d_flag_data_ready_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Accelerometer new data available.[get].
- float_t i3g4250d_from_fs245dps_to_mdps (int16_t lsb)
- float_t i3g4250d_from_lsb_to_celsius (int16_t lsb)
- int32_t i3g4250d_full_scale_get (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t ∗val)

  Gyroscope full-scale selection.[get].
- int32_t i3g4250d_full_scale_set (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t val)

  Gyroscope full-scale selection.[set].
- int32_t i3g4250d_hp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t ∗val)

  High-pass filter bandwidth selection.[get].
- int32_t i3g4250d_hp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t val)

  High-pass filter bandwidth selection.[set].
- int32_t i3g4250d_hp_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t ∗val)

  High-pass filter mode selection. [get].
- int32_t i3g4250d_hp_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t val)

  High-pass filter mode selection. [set].
- int32_t i3g4250d_hp_reference_value_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reference value for high-pass filter.[get].
- int32_t i3g4250d_hp_reference_value_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reference value for high-pass filter.[set].
- int32_t i3g4250d_int_notification_get (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t ∗val)

  Latched/pulsed interrupt.[get].
- int32_t i3g4250d_int_notification_set (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t val)

  Latched/pulsed interrupt.[set].
- int32_t i3g4250d_int_on_threshold_conf_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[get].
- int32_t i3g4250d_int_on_threshold_conf_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[set].
- int32_t i3g4250d_int_on_threshold_dur_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Durationvalue.[get].
- int32_t i3g4250d_int_on_threshold_dur_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Durationvalue.[set].
- int32_t i3g4250d_int_on_threshold_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t ∗val)

  AND/OR combination of interrupt events.[get].
- int32_t i3g4250d_int_on_threshold_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t val)

  AND/OR combination of interrupt events.[set].
- int32_t i3g4250d_int_on_threshold_src_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_src_t ∗val)

  int_on_threshold_src: [get]
- int32_t i3g4250d_int_x_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on X.[get].
- int32_t i3g4250d_int_x_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on X.[set].

- int32_t i3g4250d_int_y_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Y.[get].
- int32_t i3g4250d_int_y_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Y.[set].
- int32_t i3g4250d_int_z_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Z.[get].
- int32_t i3g4250d_int_z_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Z.[set].
- int32_t i3g4250d_lp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t ∗val)

  Lowpass filter bandwidth selection.[get].
- int32_t i3g4250d_lp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t val)

  Lowpass filter bandwidth selection.[set].
- int32_t i3g4250d_pin_int1_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t ∗val)

  Select the signal that need to route on int1 pad.[get].
- int32_t i3g4250d_pin_int1_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t val)

  Select the signal that need to route on int1 pad.[set].
- int32_t i3g4250d_pin_int2_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t ∗val)

  Select the signal that need to route on int2 pad.[get].
- int32_t i3g4250d_pin_int2_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t val)

  Select the signal that need to route on int2 pad.[set].
- int32_t i3g4250d_pin_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t ∗val)

  Push-pull/open drain selection on interrupt pads.[get].
- int32_t i3g4250d_pin_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t val)

  Push-pull/open drain selection on interrupt pads.[set].
- int32_t i3g4250d_pin_polarity_get (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t ∗val)

  Pin active-high/low.[get].
- int32_t i3g4250d_pin_polarity_set (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t val)

  Pin active-high/low.[set].
- int32_t i3g4250d_read_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Read generic device register.
- int32_t i3g4250d_self_test_get (const stmdev_ctx_t ∗ctx, i3g4250d_st_t ∗val)

  Angular rate sensor self-test enable. [get].
- int32_t i3g4250d_self_test_set (const stmdev_ctx_t ∗ctx, i3g4250d_st_t val)

  Angular rate sensor self-test enable. [set].
- int32_t i3g4250d_spi_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t ∗val)

  SPI Serial Interface Mode selection.[get].
- int32_t i3g4250d_spi_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t val)

  SPI Serial Interface Mode selection.[set].
- int32_t i3g4250d_status_reg_get (const stmdev_ctx_t ∗ctx, i3g4250d_status_reg_t ∗val)

  The STATUS_REG register is read by the primary interface.[get].
- int32_t i3g4250d_temperature_raw_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Temperature data.[get].
- int32_t i3g4250d_write_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Write generic device register.

4.1.1 Detailed Description

This file provides a set of functions needed to drive the i3g4250d enhanced inertial module.

4.1.2 Macro Definition Documentation

4.1.2.1 __weak

```
#define __weak __attribute__((weak))
```

4.1.2.2 I3G4250D_CTRL_REG1

```
#define I3G4250D_CTRL_REG1 0x20U
```

4.1.2.3 I3G4250D_CTRL_REG2

```
#define I3G4250D_CTRL_REG2 0x21U
```

4.1.2.4 I3G4250D_CTRL_REG3

```
#define I3G4250D_CTRL_REG3 0x22U
```

4.1.2.5 I3G4250D_CTRL_REG4

```
#define I3G4250D_CTRL_REG4 0x23U
```

4.1.2.6 I3G4250D_CTRL_REG5

```
#define I3G4250D_CTRL_REG5 0x24U
```

4.1.2.7 I3G4250D_FIFO_CTRL_REG

```
#define I3G4250D_FIFO_CTRL_REG 0x2EU
```

4.1.2.8 I3G4250D_FIFO_SRC_REG

```
#define I3G4250D_FIFO_SRC_REG 0x2FU
```

4.1.2.9 I3G4250D_INT1_CFG

```
#define I3G4250D_INT1_CFG 0x30U
```

4.1.2.10 I3G4250D_INT1_DURATION

```
#define I3G4250D_INT1_DURATION 0x38U
```

4.1.2.11 I3G4250D_INT1_SRC

```
#define I3G4250D_INT1_SRC 0x31U
```

4.1.2.12 I3G4250D_INT1_TSH_XH

```
#define I3G4250D_INT1_TSH_XH 0x32U
```

4.1.2.13 I3G4250D_INT1_TSH_XL

```
#define I3G4250D_INT1_TSH_XL 0x33U
```

4.1.2.14 I3G4250D_INT1_TSH_YH

```
#define I3G4250D_INT1_TSH_YH 0x34U
```

4.1.2.15 I3G4250D_INT1_TSH_YL

```
#define I3G4250D_INT1_TSH_YL 0x35U
```

4.1.2.16 I3G4250D_INT1_TSH_ZH

```
#define I3G4250D_INT1_TSH_ZH 0x36U
```

4.1.2.17 I3G4250D_INT1_TSH_ZL

```
#define I3G4250D_INT1_TSH_ZL 0x37U
```

4.1.2.18 I3G4250D_OUT_TEMP

```
#define I3G4250D_OUT_TEMP 0x26U
```

4.1.2.19 I3G4250D_OUT_X_H

```
#define I3G4250D_OUT_X_H 0x29U
```

4.1.2.20 I3G4250D_OUT_X_L

```
#define I3G4250D_OUT_X_L 0x28U
```

4.1.2.21 I3G4250D_OUT_Y_H

```
#define I3G4250D_OUT_Y_H 0x2BU
```

4.1.2.22 I3G4250D_OUT_Y_L

```
#define I3G4250D_OUT_Y_L 0x2AU
```

4.1.2.23 I3G4250D_OUT_Z_H

```
#define I3G4250D_OUT_Z_H 0x2DU
```

4.1.2.24 I3G4250D_OUT_Z_L

```
#define I3G4250D_OUT_Z_L 0x2CU
```

4.1.2.25 I3G4250D_REFERENCE

```
#define I3G4250D_REFERENCE 0x25U
```

4.1.2.26 I3G4250D_STATUS_REG

```
#define I3G4250D_STATUS_REG 0x27U
```

4.1.2.27 I3G4250D_WHO_AM_I

```
#define I3G4250D_WHO_AM_I 0x0FU
```

4.1.3 Enumeration Type Documentation

4.1.3.1 i3g4250d_and_or_t

```
enum i3g4250d_and_or_t
```

Enumerator

I3G4250D_INT1_ON_TH_AND	
I3G4250D_INT1_ON_TH_OR	

4.1.3.2 i3g4250d_ble_t

```
enum i3g4250d_ble_t
```

Enumerator

I3G4250D_AUX_LSB_AT_LOW_ADD	
I3G4250D_AUX_MSB_AT_LOW_ADD	

4.1.3.3 i3g4250d_bw_t

```
enum i3g4250d_bw_t
```

Enumerator

I3G4250D_CUT_OFF_LOW	
I3G4250D_CUT_OFF_MEDIUM	
I3G4250D_CUT_OFF_HIGH	
I3G4250D_CUT_OFF_VERY_HIGH	

4.1.3.4 i3g4250d_dr_t

```
enum i3g4250d_dr_t
```

Enumerator

I3G4250D_ODR_OFF	
I3G4250D_ODR_SLEEP	
I3G4250D_ODR_100Hz	
I3G4250D_ODR_200Hz	
I3G4250D_ODR_400Hz	
I3G4250D_ODR_800Hz	

4.1.3.5 i3g4250d_fifo_mode_t

```
enum i3g4250d_fifo_mode_t
```

Enumerator

I3G4250D_FIFO_BYPASS_MODE	
I3G4250D_FIFO_MODE	
I3G4250D_FIFO_STREAM_MODE	

4.1.3.6 i3g4250d_fs_t

```
enum i3g4250d_fs_t
```

Enumerator

I3G4250D_245dps	
I3G4250D_500dps	
I3G4250D_2000dps	

4.1.3.7 i3g4250d_h_lactive_t

```
enum i3g4250d_h_lactive_t
```

Enumerator

I3G4250D_ACTIVE_HIGH	
I3G4250D_ACTIVE_LOW	

4.1.3.8 i3g4250d_hpcf_t

```
enum i3g4250d_hpcf_t
```

Enumerator

I3G4250D_HP_LEVEL↩ _0	
I3G4250D_HP_LEVEL↩ _1	
I3G4250D_HP_LEVEL↩ _2	
I3G4250D_HP_LEVEL↩ _3	
I3G4250D_HP_LEVEL↩ _4	
I3G4250D_HP_LEVEL↩ _5	
I3G4250D_HP_LEVEL↩ _6	
I3G4250D_HP_LEVEL↩ _7	
I3G4250D_HP_LEVEL↩ _8	
I3G4250D_HP_LEVEL↩ _9	

4.1.3.9 i3g4250d_hpm_t

enum i3g4250d_hpm_t

Enumerator

I3G4250D_HP_NORMAL_MODE_WITH_RST	
I3G4250D_HP_REFERENCE_SIGNAL	
I3G4250D_HP_NORMAL_MODE	
I3G4250D_HP_AUTO_RESET_ON_INT	

4.1.3.10 i3g4250d_int1_sel_t

enum i3g4250d_int1_sel_t

Enumerator

I3G4250D_ONLY_LPF1_ON_INT	
I3G4250D_LPF1_HP_ON_INT	
I3G4250D_LPF1_LPF2_ON_INT	
I3G4250D_LPF1_HP_LPF2_ON_INT	

4.1.3.11 i3g4250d_lir_t

enum i3g4250d_lir_t

Enumerator

I3G4250D_INT_PULSED	
I3G4250D_INT_LATCHED	

4.1.3.12 i3g4250d_out_sel_t

enum i3g4250d_out_sel_t

Enumerator

I3G4250D_ONLY_LPF1_ON_OUT	
I3G4250D_LPF1_HP_ON_OUT	
I3G4250D_LPF1_LPF2_ON_OUT	
I3G4250D_LPF1_HP_LPF2_ON_OUT	

4.1.3.13 i3g4250d_pp_od_t

enum i3g4250d_pp_od_t

Enumerator

I3G4250D_PUSH_PULL	
I3G4250D_OPEN_DRAIN	

4.1.3.14 i3g4250d_sim_t

```
enum i3g4250d_sim_t
```

Enumerator

I3G4250D_SPI_4_WIRE	
I3G4250D_SPI_3_WIRE	

4.1.3.15 i3g4250d_st_t

```
enum i3g4250d_st_t
```

Enumerator

I3G4250D_GY_ST_DISABLE	
I3G4250D_GY_ST_POSITIVE	
I3G4250D_GY_ST_NEGATIVE	

4.1.4 Function Documentation

4.1.4.1 i3g4250d_angular_rate_raw_get()

```
int32_t i3g4250d_angular_rate_raw_get (
            const stmdev_ctx_t * ctx,
            int16_t * val )
```

Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

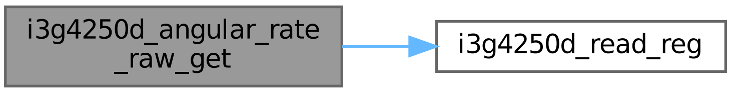
Return values

Interface	status (MANDATORY: return 0 -> no Error)

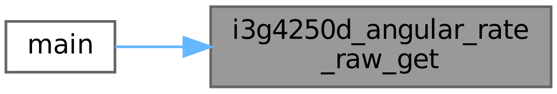
References I3G4250D_OUT_X_L, and i3g4250d_read_reg().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.2 i3g4250d_axis_x_data_get()

```
int32_t i3g4250d_axis_x_data_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

4.1.4.3 i3g4250d_axis_x_data_set()

```
int32_t i3g4250d_axis_x_data_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

4.1.4.4 i3g4250d_axis_y_data_get()

```
int32_t i3g4250d_axis_y_data_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

4.1.4.5 i3g4250d_axis_y_data_set()

```
int32_t i3g4250d_axis_y_data_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

4.1.4.6 i3g4250d_axis_z_data_get()

```
int32_t i3g4250d_axis_z_data_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

4.1.4.7 i3g4250d_axis_z_data_set()

```
int32_t i3g4250d_axis_z_data_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

4.1.4.8 i3g4250d_boot_get()

```
int32_t i3g4250d_boot_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Reboot memory content. Reload the calibration parameters.[get].

Parameters

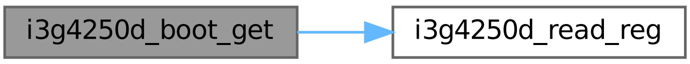
ctx	Read / write interface definitions.(ptr)
val	Get the values of boot in reg CTRL_REG5.(ptr)

Return values

interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::boot, I3G4250D_CTRL_REG5, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.9 i3g4250d_boot_set()

```
int32_t i3g4250d_boot_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Reboot memory content. Reload the calibration parameters.[set].

Parameters

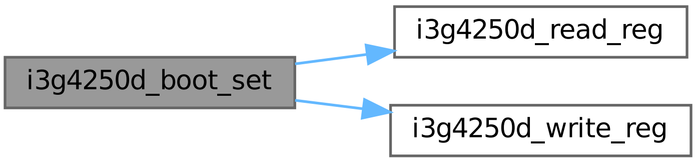
ctx	Read / write interface definitions.(ptr)
val	Change the values of boot in reg CTRL_REG5.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::boot, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.10 i3g4250d_data_format_get()

```
int32_t i3g4250d_data_format_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_ble_t * val )
```

Big/Little Endian data selection.[get].

Parameters

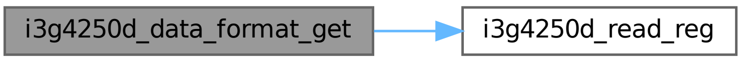
ctx	Read / write interface definitions.(ptr)
val	Get the values of "ble" in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::ble, I3G4250D_AUX_LSB_AT_LOW_ADD, I3G4250D_AUX_MSB_AT_LOW_ADD, I3G4250D_CTRL_REG4, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.11 i3g4250d_data_format_set()

```
int32_t i3g4250d_data_format_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_ble_t val )
```

Big/Little Endian data selection.[set].

Parameters

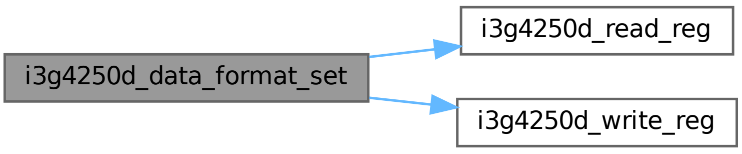
ctx	Read / write interface definitions.(ptr)
val	Change the values of "ble" in reg CTRL_REG4.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::ble, I3G4250D_CTRL_REG4, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.12 i3g4250d_data_rate_get()

```
int32_t i3g4250d_data_rate_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_dr_t * val )
```

Accelerometer data rate selection.[get].

Parameters

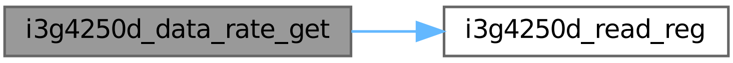
ctx	Read / write interface definitions.(ptr)
val	Get the values of dr in reg CTRL_REG1.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::dr, I3G4250D_CTRL_REG1, I3G4250D_ODR_100Hz, I3G4250D_ODR_200Hz, I3G4250D_ODR_400Hz, I3G4250D_ODR_800Hz, I3G4250D_ODR_OFF, I3G4250D_ODR_SLEEP, i3g4250d_read_reg(), and i3g4250d_ctrl_reg1_t::pd.

Here is the call graph for this function:



4.1.4.13 i3g4250d_data_rate_set()

```
int32_t i3g4250d_data_rate_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_dr_t val )
```

Accelerometer data rate selection.[set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of dr in reg CTRL_REG1

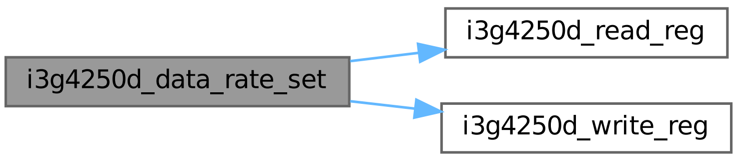
Return values

Interface	status (MANDATORY: return 0 -> no Error)

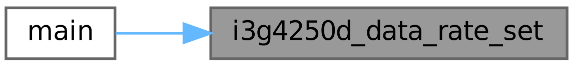
References i3g4250d_ctrl_reg1_t::dr, I3G4250D_CTRL_REG1, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg1_t::pd.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.14 i3g4250d_device_id_get()

```
int32_t i3g4250d_device_id_get (
              const stmdev_ctx_t * ctx,
              uint8_t * buff )
```

Device Who amI.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

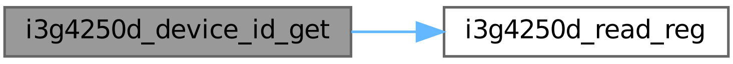
Return values

Interface	status (MANDATORY: return 0 -> no Error)

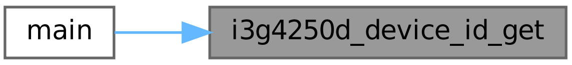
References i3g4250d_read_reg(), and I3G4250D_WHO_AM_I.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.15 i3g4250d_fifo_data_level_get()

```
int32_t i3g4250d_fifo_data_level_get (
              const stmdev_ctx_t * ctx,
              uint8_t * val )
```

FIFO stored data level[get].

Parameters

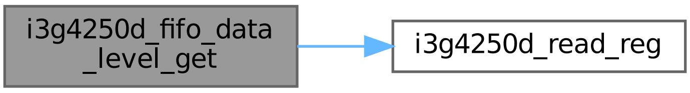
ctx	Read / write interface definitions.(ptr)
val	Get the values of fss in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_src_reg_t::fss, I3G4250D_FIFO_SRC_REG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.16 i3g4250d_fifo_empty_flag_get()

```
int32_t i3g4250d_fifo_empty_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFOemptybit.[get].

Parameters

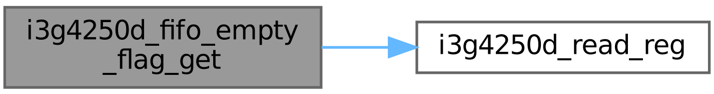
ctx	Read / write interface definitions.(ptr)
val	Get the values of empty in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_src_reg_t::empty, I3G4250D_FIFO_SRC_REG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.17 i3g4250d_fifo_enable_get()

```
int32_t i3g4250d_fifo_enable_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFOenable.[get].

Parameters

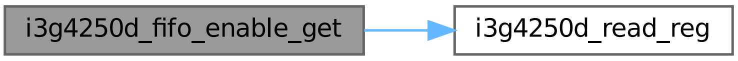
ctx	Read / write interface definitions.(ptr)
val	Get the values of fifo_en in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::fifo_en, I3G4250D_CTRL_REG5, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.18 i3g4250d_fifo_enable_set()

```
int32_t i3g4250d_fifo_enable_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

FIFOenable.[set].

Parameters

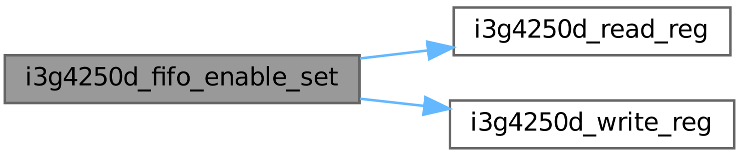
ctx	Read / write interface definitions.(ptr)
val	Change the values of fifo_en in reg CTRL_REG5

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::fifo_en, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.19 i3g4250d_fifo_mode_get()

```
int32_t i3g4250d_fifo_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_fifo_mode_t * val )
```

FIFO mode selection.[get].

Parameters

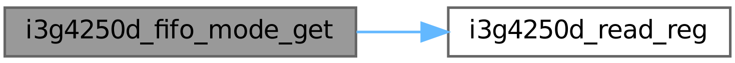
ctx	Read / write interface definitions.(ptr)
val	Get the values of fm in reg FIFO_CTRL_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_ctrl_reg_t::fm, I3G4250D_FIFO_BYPASS_MODE, I3G4250D_FIFO_CTRL_REG, I3G4250D_FIFO_MODE, I3G4250D_FIFO_STREAM_MODE, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.20 i3g4250d_fifo_mode_set()

```
int32_t i3g4250d_fifo_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_fifo_mode_t val )
```

FIFO mode selection.[set].

Parameters

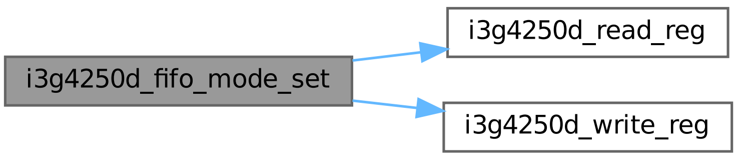
ctx	Read / write interface definitions.(ptr)
val	Change the values of fm in reg FIFO_CTRL_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_ctrl_reg_t::fm, I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.21 i3g4250d_fifo_ovr_flag_get()

```
int32_t i3g4250d_fifo_ovr_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Overrun bit status.[get].

Parameters

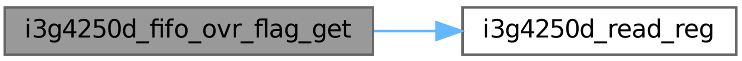
ctx	Read / write interface definitions.(ptr)
val	Get the values of ovrn in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_SRC_REG, i3g4250d_read_reg(), and i3g4250d_fifo_src_reg_t::ovrn.

Here is the call graph for this function:



4.1.4.22 i3g4250d_fifo_watermark_get()

```
int32_t i3g4250d_fifo_watermark_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFO watermark level selection.[get].

Parameters

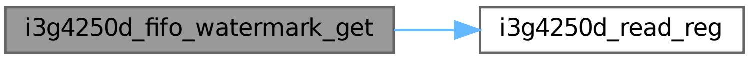
ctx	Read / write interface definitions.(ptr)
val	Get the values of wtm in reg FIFO_CTRL_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), and i3g4250d_fifo_ctrl_reg_t::wtm.

Here is the call graph for this function:



4.1.4.23 i3g4250d_fifo_watermark_set()

```
int32_t i3g4250d_fifo_watermark_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

FIFO watermark level selection.[set].

Parameters

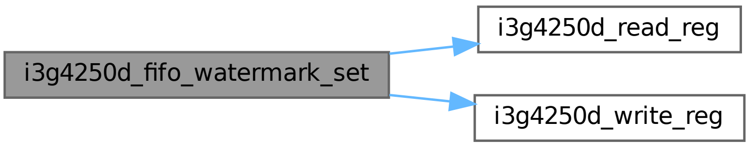
ctx	Read / write interface definitions.(ptr)
val	Change the values of wtm in reg FIFO_CTRL_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_fifo_ctrl_reg_t::wtm.

Here is the call graph for this function:



4.1.4.24 i3g4250d_fifo_wtm_flag_get()

```
int32_t i3g4250d_fifo_wtm_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)

Parameters

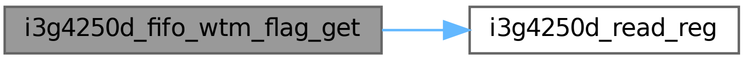
ctx	Read / write interface definitions.(ptr)
val	Get the values of wtm in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_SRC_REG, i3g4250d_read_reg(), and i3g4250d_fifo_src_reg_t::wtm.

Here is the call graph for this function:



4.1.4.25 i3g4250d_filter_path_get()

```
int32_t i3g4250d_filter_path_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_out_sel_t * val )
```

Out/FIFO selection path. [get].

Parameters

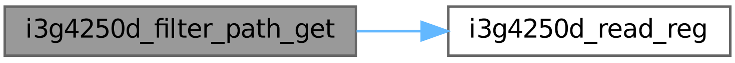
ctx	Read / write interface definitions.(ptr)
val	Get the values of out_sel in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, I3G4250D_LPF1_HP_LPF2_ON_OUT, I3G4250D_LPF1_HP_ON_OUT, I3G4250D_LPF1_LPF2_ON_OUT, I3G4250D_ONLY_LPF1_ON_OUT, i3g4250d_read_reg(), and i3g4250d_ctrl_reg5_t::out_sel.

Here is the call graph for this function:



4.1.4.26 i3g4250d_filter_path_internal_get()

```
int32_t i3g4250d_filter_path_internal_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_sel_t * val )
```

Interrupt generator selection path.[get].

Parameters

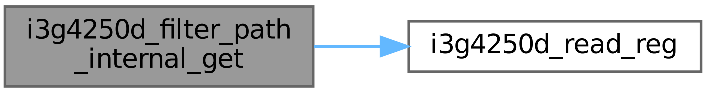
ctx	Read / write interface definitions.(ptr)
val	Get the values of int1_sel in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, I3G4250D_LPF1_HP_LPF2_ON_INT, I3G4250D_LPF1_HP_ON_INT, I3G4250D_LPF1_LPF2_ON_INT, I3G4250D_ONLY_LPF1_ON_INT, i3g4250d_read_reg(), and i3g4250d_ctrl_reg5_t::int1_sel.

Here is the call graph for this function:



4.1.4.27 i3g4250d_filter_path_internal_set()

```
int32_t i3g4250d_filter_path_internal_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_sel_t val )
```

Interrupt generator selection path.[set].

Parameters

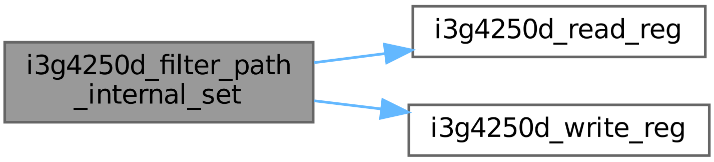
ctx	Read / write interface definitions.(ptr)
val	Change the values of int1_sel in reg CTRL_REG5

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg5_t::int1_sel.

Here is the call graph for this function:



4.1.4.28 i3g4250d_filter_path_set()

```
int32_t i3g4250d_filter_path_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_out_sel_t val )
```

Out/FIFO selection path. [set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "out_sel" in reg CTRL_REG5.

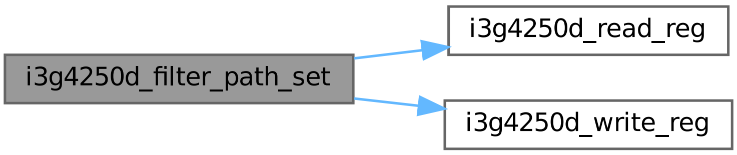
Return values

Interface	status (MANDATORY: return 0 -> no Error)

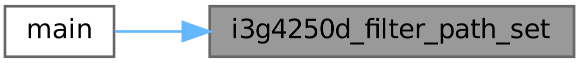
References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg5_t::out_sel.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.29 i3g4250d_flag_data_ready_get()

```
int32_t i3g4250d_flag_data_ready_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Accelerometer new data available.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "zyxda" in reg STATUS_REG.(ptr)

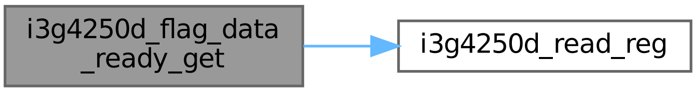
Return values

Interface	status (MANDATORY: return 0 -> no Error)

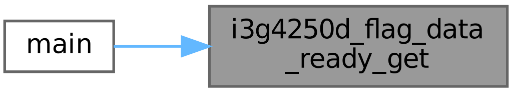
References i3g4250d_read_reg(), I3G4250D_STATUS_REG, and i3g4250d_status_reg_t::zyxda.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

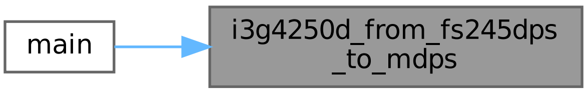


4.1.4.30 i3g4250d_from_fs245dps_to_mdps()

```
float_t i3g4250d_from_fs245dps_to_mdps (
             int16_t lsb )
```

Referenced by main().

Here is the caller graph for this function:



4.1.4.31 i3g4250d_from_lsb_to_celsius()

```
float_t i3g4250d_from_lsb_to_celsius (
             int16_t lsb )
```

4.1.4.32 i3g4250d_full_scale_get()

```
int32_t i3g4250d_full_scale_get (
             const stmdev_ctx_t * ctx,
             i3g4250d_fs_t * val )
```

Gyroscope full-scale selection.[get].

Parameters

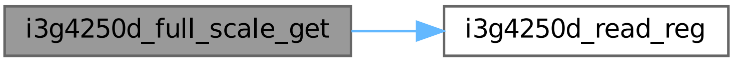
ctx	read / write interface definitions(ptr)
val	Get the values of fs in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::fs, I3G4250D_2000dps, I3G4250D_245dps, I3G4250D_500dps, I3G4250D_CTRL_REG4, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.33 i3g4250d_full_scale_set()

```
int32_t i3g4250d_full_scale_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_fs_t val )
```

Gyroscope full-scale selection.[set].

Parameters

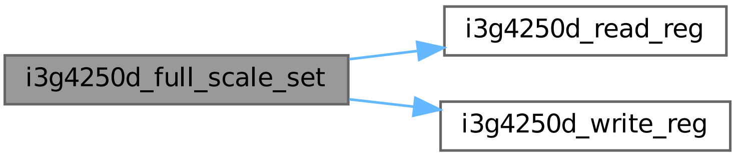
ctx	read / write interface definitions(ptr)
val	change the values of fs in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::fs, I3G4250D_CTRL_REG4, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.34 i3g4250d_hp_bandwidth_get()

```
int32_t i3g4250d_hp_bandwidth_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpcf_t * val )
```

High-pass filter bandwidth selection.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of hpcf in reg CTRL_REG2.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpcf, I3G4250D_CTRL_REG2, I3G4250D_HP_LEVEL_0, I3G4250D_HP_LEVEL_1, I3G4250D_HP_LEVEL_2, I3G4250D_HP_LEVEL_3, I3G4250D_HP_LEVEL_4, I3G4250D_HP_LEVEL_5, I3G4250D_HP_LEVEL_6, I3G4250D_HP_LEVEL_7, I3G4250D_HP_LEVEL_8, I3G4250D_HP_LEVEL_9, and i3g4250d_read_reg().

Here is the call graph for this function:

i3g4250d_hp_bandwidth_get
i3g4250d_read_reg

4.1.4.35 i3g4250d_hp_bandwidth_set()

```
int32_t i3g4250d_hp_bandwidth_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpcf_t val )
```

High-pass filter bandwidth selection.[set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "hpcf" in reg CTRL_REG2.

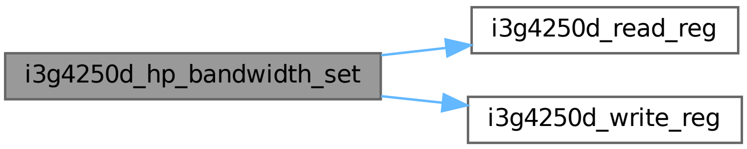
Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpcf, I3G4250D_CTRL_REG2, i3g4250d_read_reg(), and i3g4250d_write_reg().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

main
i3g4250d_hp_bandwidth_set

4.1.4.36 i3g4250d_hp_mode_get()

```
int32_t i3g4250d_hp_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpm_t * val )
```

High-pass filter mode selection. [get].

Parameters

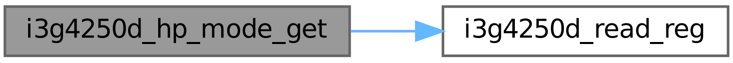
ctx	Read / write interface definitions.(ptr)
val	Get the values of hpm in reg CTRL_REG2.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpm, I3G4250D_CTRL_REG2, I3G4250D_HP_AUTO_RESET_ON_INT, I3G4250D_HP_NORMAL_MODE, I3G4250D_HP_NORMAL_MODE_WITH_RST, I3G4250D_HP_REFERENCE_SIGNAL, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.37 i3g4250d_hp_mode_set()

```
int32_t i3g4250d_hp_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpm_t val )
```

High-pass filter mode selection. [set].

Parameters

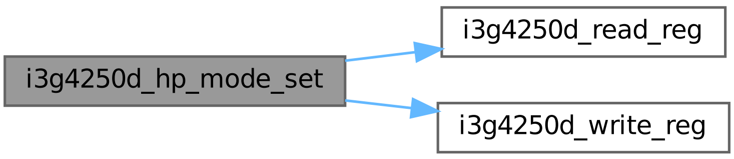
ctx	Read / write interface definitions.(ptr)
val	Change the values of "hpm" in reg CTRL_REG2.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpm, I3G4250D_CTRL_REG2, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.38 i3g4250d_hp_reference_value_get()

```
int32_t i3g4250d_hp_reference_value_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Reference value for high-pass filter.[get].

Parameters

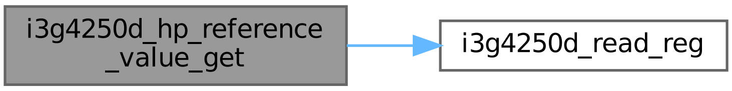
ctx	Read / write interface definitions.(ptr)
val	Get the values of ref in reg REFERENCE.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), I3G4250D_REFERENCE, and i3g4250d_reference_t::ref.

Here is the call graph for this function:



4.1.4.39 i3g4250d_hp_reference_value_set()

```
int32_t i3g4250d_hp_reference_value_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Reference value for high-pass filter.[set].

Parameters

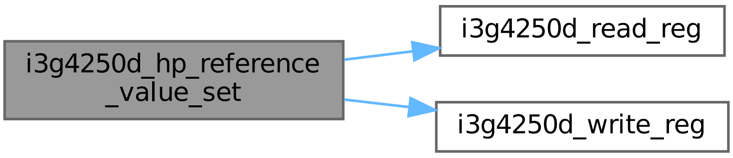
ctx	Read / write interface definitions.(ptr)
val	Change the values of ref in reg REFERENCE

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), I3G4250D_REFERENCE, i3g4250d_write_reg(), and i3g4250d_reference_t::ref.

Here is the call graph for this function:



4.1.4.40 i3g4250d_int_notification_get()

```
int32_t i3g4250d_int_notification_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_lir_t * val )
```

Latched/pulsed interrupt.[get].

Parameters

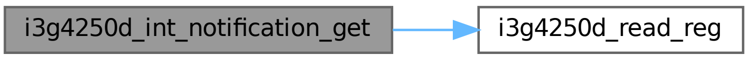
ctx	Read / write interface definitions.(ptr)
val	Get the values of lir in reg INT1_CFG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, I3G4250D_INT_LATCHED, I3G4250D_INT_PULSED, i3g4250d_read_reg(), and i3g4250d_int1_cfg_t::lir.

Here is the call graph for this function:



4.1.4.41 i3g4250d_int_notification_set()

```
int32_t i3g4250d_int_notification_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_lir_t val )
```

Latched/pulsed interrupt.[set].

Parameters

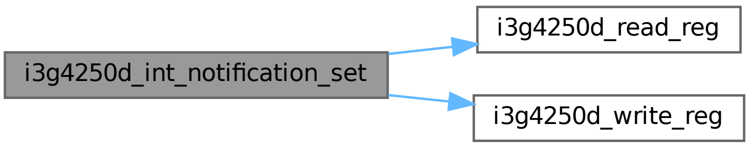
ctx	Read / write interface definitions.(ptr)
val	Change the values of lir in reg INT1_CFG.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_cfg_t::lir.

Here is the call graph for this function:



4.1.4.42 i3g4250d_int_on_threshold_conf_get()

```
int32_t i3g4250d_int_on_threshold_conf_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_cfg_t * val )
```

Configure the interrupt threshold sign.[get].

Parameters

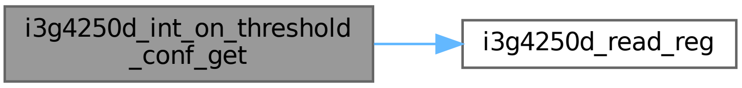
ctx	Read / write interface definitions.(ptr)
val	Struct of registers from INT1_CFG to.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.43 i3g4250d_int_on_threshold_conf_set()

```
int32_t i3g4250d_int_on_threshold_conf_set (
              const stmdev_ctx_t * ctx,
              i3g4250d_int1_cfg_t * val )
```

Configure the interrupt threshold sign.[set].

Parameters

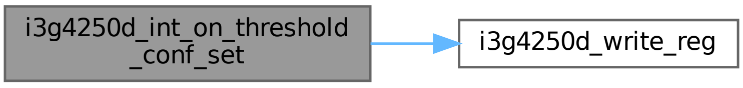
ctx	Read / write interface definitions.(ptr)
val	Struct of registers INT1_CFG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.44 i3g4250d_int_on_threshold_dur_get()

```
int32_t i3g4250d_int_on_threshold_dur_get (
              const stmdev_ctx_t * ctx,
              uint8_t * val )
```

Durationvalue.[get].

Parameters

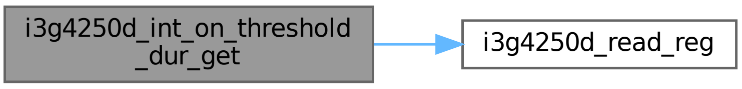
ctx	Read / write interface definitions.(ptr)
val	Get the values of d in reg INT1_DURATION.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_duration_t::d, I3G4250D_INT1_DURATION, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.45 i3g4250d_int_on_threshold_dur_set()

```
int32_t i3g4250d_int_on_threshold_dur_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Durationvalue.[set].

Parameters

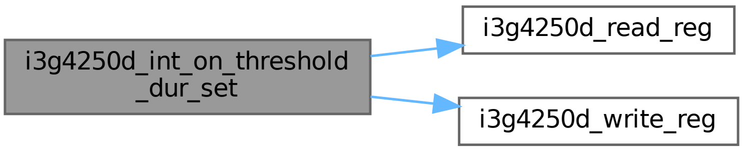
ctx	Read / write interface definitions.(ptr)
val	Change the values of d in reg INT1_DURATION

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_duration_t::d, I3G4250D_INT1_DURATION, i3g4250d_read_reg(), i3g4250d_write_reg(), PROPERTY_DISABLE, PROPERTY_ENABLE, and i3g4250d_int1_duration_t::wait.

Here is the call graph for this function:



4.1.4.46 i3g4250d_int_on_threshold_mode_get()

```
int32_t i3g4250d_int_on_threshold_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_and_or_t * val )
```

AND/OR combination of interrupt events.[get].

Parameters

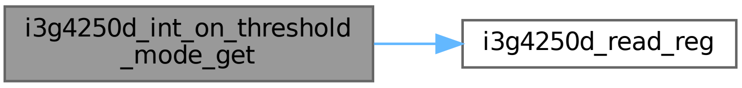
ctx	Read / write interface definitions.(ptr)
val	Get the values of and_or in reg INT1_CFG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_cfg_t::and_or, I3G4250D_INT1_CFG, I3G4250D_INT1_ON_TH_AND, I3G4250D_INT1_ON_TH_OR, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.47 i3g4250d_int_on_threshold_mode_set()

```
int32_t i3g4250d_int_on_threshold_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_and_or_t val )
```

AND/OR combination of interrupt events.[set].

Parameters

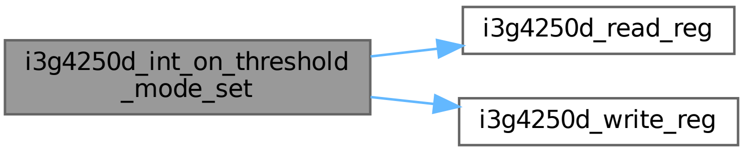
ctx	Read / write interface definitions.(ptr)
val	Change the values of and_or in reg INT1_CFG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_cfg_t::and_or, I3G4250D_INT1_CFG, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.48 i3g4250d_int_on_threshold_src_get()

```
int32_t i3g4250d_int_on_threshold_src_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_src_t * val )
```

int_on_threshold_src: [get]

Parameters

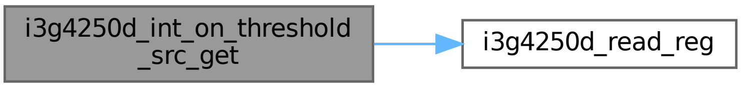
ctx	Read / write interface definitions.(ptr)
val	Union of registers from INT1_SRC to.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_SRC, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.49 i3g4250d_int_x_threshold_get()

```
int32_t i3g4250d_int_x_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on X.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of thsx in reg INT1_TSH_XH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_XH, I3G4250D_INT1_TSH_XL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_xh_t::thsx.

Here is the call graph for this function:

i3g4250d_int_x_threshold_get
i3g4250d_read_reg

4.1.4.50 i3g4250d_int_x_threshold_set()

```
int32_t i3g4250d_int_x_threshold_set (
            const stmdev_ctx_t * ctx,
            uint16_t val )
```

Interrupt threshold on X.[set].

Parameters

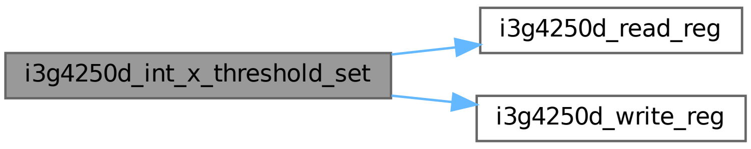
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsx in reg INT1_TSH_XH

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_XH, I3G4250D_INT1_TSH_XL, i3g4250d_read_reg(), i3g4250d_write_reg(), i3g4250d_int1_tsh_xh_t::thsx, and i3g4250d_int1_tsh_xl_t::thsx.

Here is the call graph for this function:



4.1.4.51 i3g4250d_int_y_threshold_get()

```
int32_t i3g4250d_int_y_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on Y.[get].

Parameters

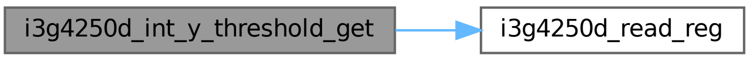
ctx	Read / write interface definitions.(ptr)
val	Get the values of thsy in reg INT1_TSH_YH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_YH, I3G4250D_INT1_TSH_YL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_yh_t::thsy.

Here is the call graph for this function:



4.1.4.52 i3g4250d_int_y_threshold_set()

```
int32_t i3g4250d_int_y_threshold_set (
            const stmdev_ctx_t * ctx,
            uint16_t val )
```

Interrupt threshold on Y.[set].

Parameters

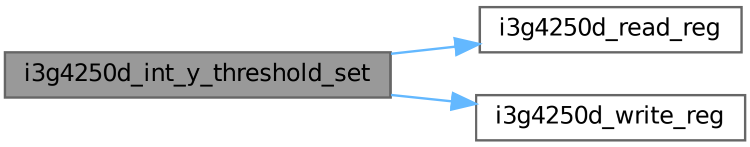
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsy in reg INT1_TSH_YH

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_YH, I3G4250D_INT1_TSH_YL, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_tsh_yh_t::thsy.

Here is the call graph for this function:



4.1.4.53 i3g4250d_int_z_threshold_get()

```
int32_t i3g4250d_int_z_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on Z.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of thsz in reg INT1_TSH_ZH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_ZH, I3G4250D_INT1_TSH_ZL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_zh_t::thsz.

Here is the call graph for this function:

i3g4250d_int_z_threshold_get
i3g4250d_read_reg

4.1.4.54 i3g4250d_int_z_threshold_set()

```
int32_t i3g4250d_int_z_threshold_set (
              const stmdev_ctx_t * ctx,
              uint16_t val )
```

Interrupt threshold on Z.[set].

Parameters

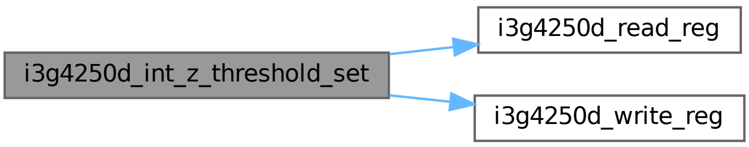
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsz in reg INT1_TSH_ZH.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_ZH, I3G4250D_INT1_TSH_ZL, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_tsh_zh_t::thsz.

Here is the call graph for this function:



4.1.4.55 i3g4250d_lp_bandwidth_get()

```
int32_t i3g4250d_lp_bandwidth_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_bw_t * val )
```

Lowpass filter bandwidth selection.[get].

Parameters

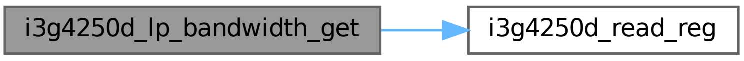
ctx	Read / write interface definitions.(ptr)
val	Get the values of "bw" in reg CTRL_REG1.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::bw, I3G4250D_CTRL_REG1, I3G4250D_CUT_OFF_HIGH, I3G4250D_CUT_OFF_LOW, I3G4250D_CUT_OFF_MEDIUM, I3G4250D_CUT_OFF_VERY_HIGH, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.56 i3g4250d_lp_bandwidth_set()

```
int32_t i3g4250d_lp_bandwidth_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_bw_t val )
```

Lowpass filter bandwidth selection.[set].

Parameters

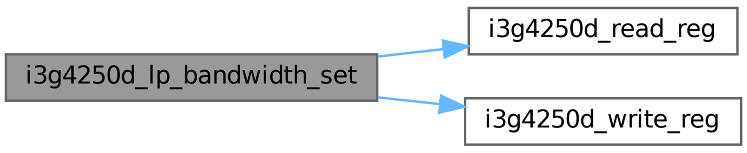
ctx	Read / write interface definitions.(ptr)
val	Change the values of "bw" in reg CTRL_REG1.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::bw, I3G4250D_CTRL_REG1, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.57 i3g4250d_pin_int1_route_get()

```
int32_t i3g4250d_pin_int1_route_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_route_t * val )
```

Select the signal that need to route on int1 pad.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Read CTRL_REG3 int1 pad.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i1_boot, i3g4250d_int1_route_t::i1_boot, i3g4250d_ctrl_reg3_t::i1_int1, i3g4250d_int1_route_t::i1_int1, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:

i3g4250d_pin_int1_route_get
i3g4250d_read_reg

4.1.4.58 i3g4250d_pin_int1_route_set()

```
int32_t i3g4250d_pin_int1_route_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_route_t val )
```

Select the signal that need to route on int1 pad.[set].

Parameters

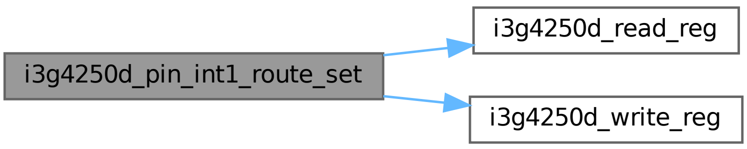
ctx	Read / write interface definitions.(ptr)
val	Configure CTRL_REG3 int1 pad

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i1_boot, i3g4250d_int1_route_t::i1_boot, i3g4250d_ctrl_reg3_t::i1_int1, i3g4250d_int1_route_t::i1_int1, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.59 i3g4250d_pin_int2_route_get()

```
int32_t i3g4250d_pin_int2_route_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int2_route_t * val )
```

Select the signal that need to route on int2 pad.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Read CTRL_REG3 int2 pad.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i2_drdy, i3g4250d_int2_route_t::i2_drdy, i3g4250d_ctrl_reg3_t::i2_empty, i3g4250d_int2_route_t::i2_empty, i3g4250d_ctrl_reg3_t::i2_orun, i3g4250d_int2_route_t::i2_orun, i3g4250d_ctrl_reg3_t::i2_wtm, i3g4250d_int2_route_t::i2_wtm, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:

i3g4250d_pin_int2_route_get
i3g4250d_read_reg

4.1.4.60 i3g4250d_pin_int2_route_set()

```
int32_t i3g4250d_pin_int2_route_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int2_route_t val )
```

Select the signal that need to route on int2 pad.[set].

Parameters

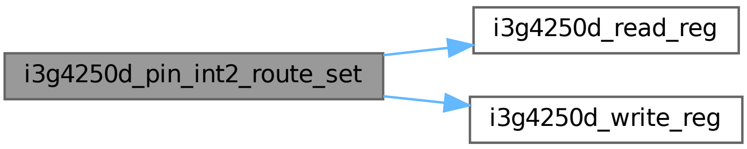
ctx	Read / write interface definitions.(ptr)
val	Configure CTRL_REG3 int2 pad

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i2_drdy, i3g4250d_int2_route_t::i2_drdy, i3g4250d_ctrl_reg3_t::i2_empty, i3g4250d_int2_route_t::i2_empty, i3g4250d_ctrl_reg3_t::i2_orun, i3g4250d_int2_route_t::i2_orun, i3g4250d_ctrl_reg3_t::i2_wtm, i3g4250d_int2_route_t::i2_wtm, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.61 i3g4250d_pin_mode_get()

```
int32_t i3g4250d_pin_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_pp_od_t * val )
```

Push-pull/open drain selection on interrupt pads.[get].

Parameters

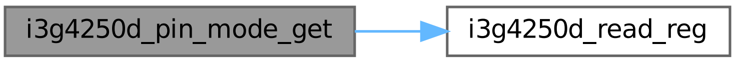
ctx	Read / write interface definitions.(ptr)
val	Get the values of pp_od in reg CTRL_REG3.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG3, I3G4250D_OPEN_DRAIN, I3G4250D_PUSH_PULL, i3g4250d_read_reg(), and i3g4250d_ctrl_reg3_t::pp_od.

Here is the call graph for this function:



4.1.4.62 i3g4250d_pin_mode_set()

```
int32_t i3g4250d_pin_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_pp_od_t val )
```

Push-pull/open drain selection on interrupt pads.[set].

Parameters

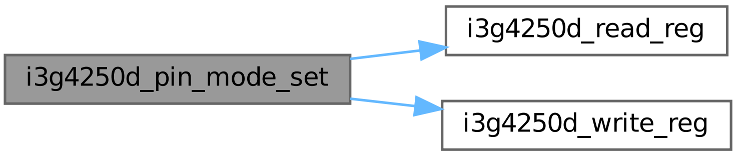
ctx	Read / write interface definitions.(ptr)
val	Change the values of pp_od in reg CTRL_REG3

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG3, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg3_t::pp_od.

Here is the call graph for this function:



4.1.4.63 i3g4250d_pin_polarity_get()

```
int32_t i3g4250d_pin_polarity_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_h_lactive_t * val )
```

Pin active-high/low.[get].

Parameters

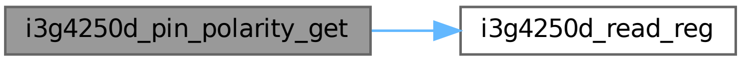
ctx	Read / write interface definitions.(ptr)
val	Get the values of h_lactive in reg CTRL_REG3.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::h_lactive, I3G4250D_ACTIVE_HIGH, I3G4250D_ACTIVE_LOW, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.64 i3g4250d_pin_polarity_set()

```
int32_t i3g4250d_pin_polarity_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_h_lactive_t val )
```

Pin active-high/low.[set].

Parameters

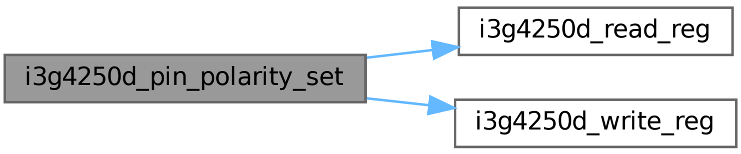
ctx	Read / write interface definitions.(ptr)
val	Change the values of h_lactive in reg CTRL_REG3.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::h_lactive, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.4.65 i3g4250d_read_reg()

```
int32_t i3g4250d_read_reg (
            const stmdev_ctx_t * ctx,
            uint8_t reg,
            uint8_t * data,
            uint16_t len )
```

Read generic device register.

Parameters

ctx	read / write interface definitions(ptr)
reg	register to read
data	pointer to buffer that store the data read(ptr)
len	number of consecutive register to read

Return values

interface	status (MANDATORY: return 0 -> no Error)

References stmdev_ctx_t::handle, and stmdev_ctx_t::read_reg.

Referenced by i3g4250d_angular_rate_raw_get(), i3g4250d_boot_get(), i3g4250d_boot_set(), i3g4250d_data_format_get(), i3g4250d_data_format_set(), i3g4250d_data_rate_get(), i3g4250d_data_rate_set(), i3g4250d_device_id_get(), i3g4250d_fifo_data_level_get(), i3g4250d_fifo_empty_flag_get(), i3g4250d_fifo_enable_get(), i3g4250d_fifo_enable_set(), i3g4250d_fifo_mode_get(), i3g4250d_fifo_mode_set(), i3g4250d_fifo_ovr_flag_get(), i3g4250d_fifo_watermark_get(), i3g4250d_fifo_watermark_set(), i3g4250d_fifo_wtm_flag_get(), i3g4250d_filter_path_get(), i3g4250d_filter_path_internal_get(), i3g4250d_filter_path_internal_set(), i3g4250d_filter_path_set(), i3g4250d_flag_data_ready_get(), i3g4250d_full_scale_get(), i3g4250d_full_scale_set(), i3g4250d_hp_bandwidth_get(), i3g4250d_hp_bandwidth_set(), i3g4250d_hp_mode_get(), i3g4250d_hp_mode_set(), i3g4250d_hp_reference_value_get(), i3g4250d_hp_reference_value_set(), i3g4250d_int_notification_get()
i3g4250d_int_notification_set(), i3g4250d_int_on_threshold_conf_get(), i3g4250d_int_on_threshold_dur_get(), i3g4250d_int_on_threshold_dur_set(), i3g4250d_int_on_threshold_mode_get(), i3g4250d_int_on_threshold_mode_set(), i3g4250d_int_on_threshold_src_get(), i3g4250d_int_x_threshold_get(), i3g4250d_int_x_threshold_set(), i3g4250d_int_y_threshold_g
i3g4250d_int_y_threshold_set(), i3g4250d_int_z_threshold_get(), i3g4250d_int_z_threshold_set(), i3g4250d_lp_bandwidth_get(), i3g4250d_lp_bandwidth_set(), i3g4250d_pin_int1_route_get(), i3g4250d_pin_int1_route_set(), i3g4250d_pin_int2_route_get(), i3g4250d_pin_int2_route_set(), i3g4250d_pin_mode_get(), i3g4250d_pin_mode_set(), i3g4250d_pin_polarity_get(), i3g4250d_pin_polarity_set(), i3g4250d_self_test_get(), i3g4250d_self_test_set(), i3g4250d_spi_mode_get(), i3g4250d_spi_mode_set(), i3g4250d_status_reg_get(), and i3g4250d_temperature_raw_get().

4.1.4.66 i3g4250d_self_test_get()

```
int32_t i3g4250d_self_test_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_st_t * val )
```

Angular rate sensor self-test enable. [get].

Parameters

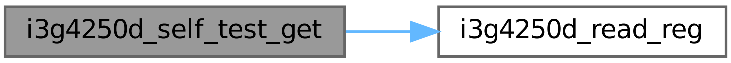
ctx	Read / write interface definitions.(ptr)
val	Get the values of st in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, I3G4250D_GY_ST_DISABLE, I3G4250D_GY_ST_NEGATIVE, I3G4250D_GY_ST_POSITIVE
i3g4250d_read_reg(), and i3g4250d_ctrl_reg4_t::st.

Here is the call graph for this function:



4.1.4.67 i3g4250d_self_test_set()

```
int32_t i3g4250d_self_test_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_st_t val )
```

Angular rate sensor self-test enable. [set].

Parameters

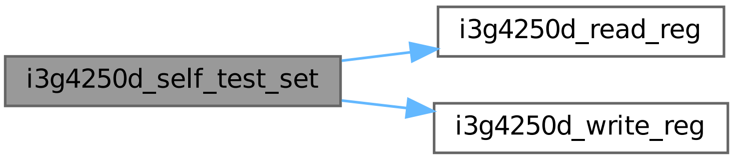
ctx	Read / write interface definitions.(ptr)
val	change the values of st in reg CTRL_REG4.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg4_t::st.

Here is the call graph for this function:



4.1.4.68 i3g4250d_spi_mode_get()

```
int32_t i3g4250d_spi_mode_get (
              const stmdev_ctx_t * ctx,
              i3g4250d_sim_t * val )
```

SPI Serial Interface Mode selection.[get].

Parameters

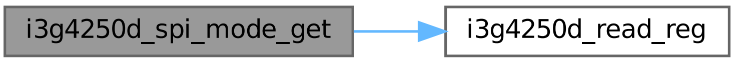
ctx	Read / write interface definitions.(ptr)
val	Get the values of sim in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), I3G4250D_SPI_3_WIRE, I3G4250D_SPI_4_WIRE, and i3g4250d_ctrl_reg4_t::sim.

Here is the call graph for this function:



4.1.4.69 i3g4250d_spi_mode_set()

```
int32_t i3g4250d_spi_mode_set (
              const stmdev_ctx_t * ctx,
              i3g4250d_sim_t val )
```

SPI Serial Interface Mode selection.[set].

Parameters

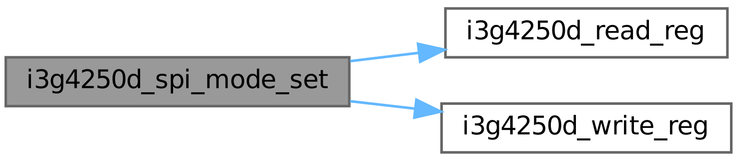
ctx	Read / write interface definitions.(ptr)
val	Change the values of sim in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg4_t::sim.

Here is the call graph for this function:



4.1.4.70 i3g4250d_status_reg_get()

```
int32_t i3g4250d_status_reg_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_status_reg_t * val )
```

The STATUS_REG register is read by the primary interface.[get].

Parameters

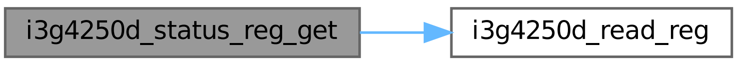
ctx	Read / write interface definitions.(ptr)
val	registers STATUS_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), and I3G4250D_STATUS_REG.

Here is the call graph for this function:



4.1.4.71 i3g4250d_temperature_raw_get()

```
int32_t i3g4250d_temperature_raw_get (
            const stmdev_ctx_t * ctx,
            uint8_t * buff )
```

Temperature data.[get].

Parameters

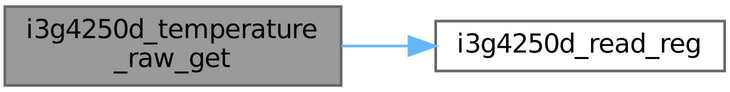
ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_OUT_TEMP, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.4.72 i3g4250d_write_reg()

```
int32_t i3g4250d_write_reg (
            const stmdev_ctx_t * ctx,
            uint8_t reg,
            uint8_t * data,
            uint16_t len )
```

Write generic device register.

Parameters

ctx	read / write interface definitions(ptr)
reg	register to write
data	pointer to data to write in register reg(ptr)
len	number of consecutive register to write

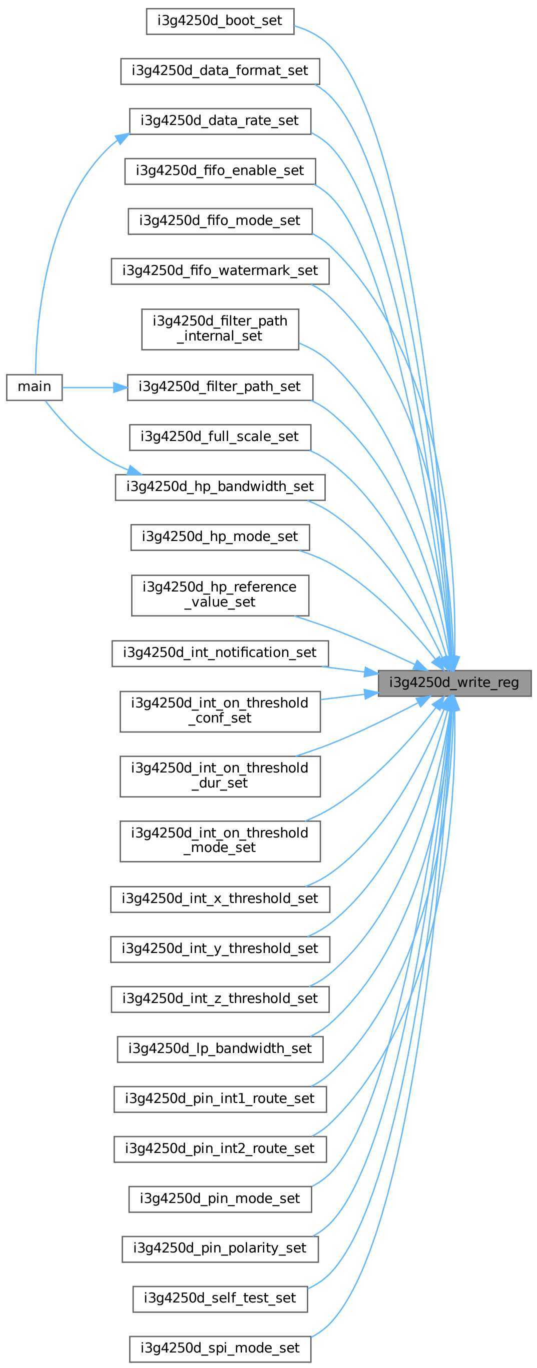
Return values

interface	status (MANDATORY: return 0 -> no Error)

References stmdev_ctx_t::handle, and stmdev_ctx_t::write_reg.

Referenced by i3g4250d_boot_set(), i3g4250d_data_format_set(), i3g4250d_data_rate_set(), i3g4250d_fifo_enable_set(), i3g4250d_fifo_mode_set(), i3g4250d_fifo_watermark_set(), i3g4250d_filter_path_internal_set(), i3g4250d_filter_path_set(), i3g4250d_full_scale_set(), i3g4250d_hp_bandwidth_set(), i3g4250d_hp_mode_set(), i3g4250d_hp_reference_value_set(), i3g4250d_int_notification_set(), i3g4250d_int_on_threshold_conf_set(), i3g4250d_int_on_threshold_dur_set(), i3g4250d_int_on_threshold_mode_set(), i3g4250d_int_x_threshold_set(), i3g4250d_int_y_threshold_set(), i3g4250d_int_z_threshold_set(), i3g4250d_lp_bandwidth_set(), i3g4250d_pin_int1_route_set(), i3g4250d_pin_int2_route_set(), i3g4250d_pin_mode_set(), i3g4250d_pin_polarity_set(), i3g4250d_self_test_set(), and i3g4250d_spi_mode_set().

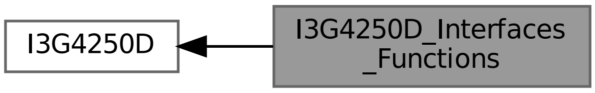
Here is the caller graph for this function:


i3g4250d_boot_set
i3g4250d_data_format_set
i3g4250d_data_rate_set
i3g4250d_fifo_enable_set
i3g4250d_fifo_mode_set
i3g4250d_fifo_watermark_set
i3g4250d_filter_path_internal_set
main
i3g4250d_filter_path_set
i3g4250d_full_scale_set
i3g4250d_hp_bandwidth_set
i3g4250d_hp_mode_set
i3g4250d_hp_reference_value_set
i3g4250d_int_notification_set
i3g4250d_write_reg
i3g4250d_int_on_threshold_conf_set
i3g4250d_int_on_threshold_dur_set
i3g4250d_int_on_threshold_mode_set
i3g4250d_int_x_threshold_set
i3g4250d_int_y_threshold_set
i3g4250d_int_z_threshold_set
i3g4250d_lp_bandwidth_set
i3g4250d_pin_int1_route_set
i3g4250d_pin_int2_route_set
i3g4250d_pin_mode_set
i3g4250d_pin_polarity_set
i3g4250d_self_test_set
i3g4250d_spi_mode_set

4.1.5 I3G4250D_Interfaces_Functions

This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.

Collaboration diagram for I3G4250D_Interfaces_Functions:



Functions

- int32_t __weak i3g4250d_read_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Read generic device register.
- int32_t __weak i3g4250d_write_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Write generic device register.

4.1.5.1 Detailed Description

This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.

4.1.5.2 Function Documentation

4.1.5.2.1 i3g4250d_read_reg()

```
int32_t __weak i3g4250d_read_reg (
            const stmdev_ctx_t * ctx,
            uint8_t reg,
            uint8_t * data,
            uint16_t len )
```

Read generic device register.

Parameters

ctx	read / write interface definitions(ptr)
reg	register to read
data	pointer to buffer that store the data read(ptr)
len	number of consecutive register to read

Return values

interface	status (MANDATORY: return 0 -> no Error)

References stmdev_ctx_t::handle, and stmdev_ctx_t::read_reg.

Referenced by i3g4250d_angular_rate_raw_get(), i3g4250d_boot_get(), i3g4250d_boot_set(), i3g4250d_data_format_get(), i3g4250d_data_format_set(), i3g4250d_data_rate_get(), i3g4250d_data_rate_set(), i3g4250d_device_id_get(), i3g4250d_fifo_data_level_get(), i3g4250d_fifo_empty_flag_get(), i3g4250d_fifo_enable_get(), i3g4250d_fifo_enable_set(), i3g4250d_fifo_mode_get(), i3g4250d_fifo_mode_set(), i3g4250d_fifo_ovr_flag_get(), i3g4250d_fifo_watermark_get(), i3g4250d_fifo_watermark_set(), i3g4250d_fifo_wtm_flag_get(), i3g4250d_filter_path_get(), i3g4250d_filter_path_internal_get(), i3g4250d_filter_path_internal_set(), i3g4250d_filter_path_set(), i3g4250d_flag_data_ready_get(), i3g4250d_full_scale_get(), i3g4250d_full_scale_set(), i3g4250d_hp_bandwidth_get(), i3g4250d_hp_bandwidth_set(), i3g4250d_hp_mode_get(), i3g4250d_hp_mode_set(), i3g4250d_hp_reference_value_get(), i3g4250d_hp_reference_value_set(), i3g4250d_int_notification_get() i3g4250d_int_notification_set(), i3g4250d_int_on_threshold_conf_get(), i3g4250d_int_on_threshold_dur_get(), i3g4250d_int_on_threshold_dur_set(), i3g4250d_int_on_threshold_mode_get(), i3g4250d_int_on_threshold_mode_set(), i3g4250d_int_on_threshold_src_get(), i3g4250d_int_x_threshold_get(), i3g4250d_int_x_threshold_set(), i3g4250d_int_y_threshold_g i3g4250d_int_y_threshold_set(), i3g4250d_int_z_threshold_get(), i3g4250d_int_z_threshold_set(), i3g4250d_lp_bandwidth_get(), i3g4250d_lp_bandwidth_set(), i3g4250d_pin_int1_route_get(), i3g4250d_pin_int1_route_set(), i3g4250d_pin_int2_route_get(), i3g4250d_pin_int2_route_set(), i3g4250d_pin_mode_get(), i3g4250d_pin_mode_set(), i3g4250d_pin_polarity_get(), i3g4250d_pin_polarity_set(), i3g4250d_self_test_get(), i3g4250d_self_test_set(), i3g4250d_spi_mode_get(), i3g4250d_spi_mode_set(), i3g4250d_status_reg_get(), and i3g4250d_temperature_raw_get().

4.1.5.2.2 i3g4250d_write_reg()

```
int32_t __weak i3g4250d_write_reg (
            const stmdev_ctx_t * ctx,
            uint8_t reg,
            uint8_t * data,
            uint16_t len )
```

Write generic device register.

Parameters

ctx	read / write interface definitions(ptr)
reg	register to write
data	pointer to data to write in register reg(ptr)
len	number of consecutive register to write

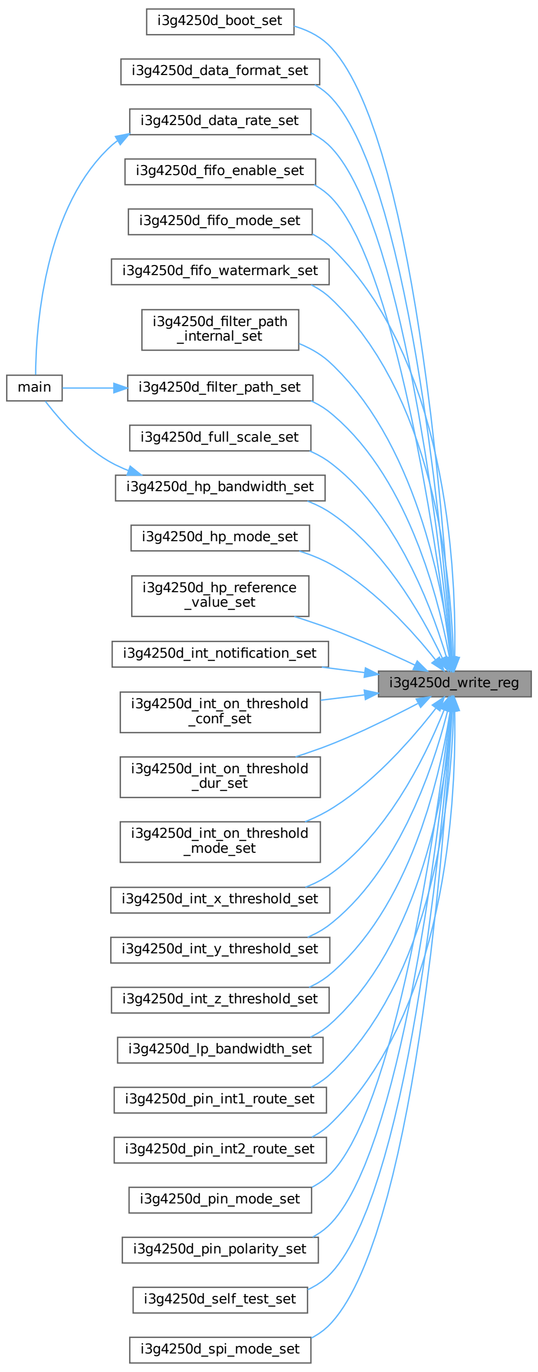
Return values

interface	status (MANDATORY: return 0 -> no Error)

References stmdev_ctx_t::handle, and stmdev_ctx_t::write_reg.

Referenced by i3g4250d_boot_set(), i3g4250d_data_format_set(), i3g4250d_data_rate_set(), i3g4250d_fifo_enable_set(), i3g4250d_fifo_mode_set(), i3g4250d_fifo_watermark_set(), i3g4250d_filter_path_internal_set(), i3g4250d_filter_path_set(), i3g4250d_full_scale_set(), i3g4250d_hp_bandwidth_set(), i3g4250d_hp_mode_set(), i3g4250d_hp_reference_value_set(), i3g4250d_int_notification_set(), i3g4250d_int_on_threshold_conf_set(), i3g4250d_int_on_threshold_dur_set(), i3g4250d_int_on_threshold_mode_set(), i3g4250d_int_x_threshold_set(), i3g4250d_int_y_threshold_set(), i3g4250d_int_z_threshold_set(), i3g4250d_lp_bandwidth_set(), i3g4250d_pin_int1_route_set(), i3g4250d_pin_int2_route_set(), i3g4250d_pin_mode_set(), i3g4250d_pin_polarity_set(), i3g4250d_self_test_set(), and i3g4250d_spi_mode_set().

Here is the caller graph for this function:


i3g4250d_boot_set
i3g4250d_data_format_set
i3g4250d_data_rate_set
i3g4250d_fifo_enable_set
i3g4250d_fifo_mode_set
i3g4250d_fifo_watermark_set
i3g4250d_filter_path_internal_set
main
i3g4250d_filter_path_set
i3g4250d_full_scale_set
i3g4250d_hp_bandwidth_set
i3g4250d_hp_mode_set
i3g4250d_hp_reference_value_set
i3g4250d_int_notification_set
i3g4250d_write_reg
i3g4250d_int_on_threshold_conf_set
i3g4250d_int_on_threshold_dur_set
i3g4250d_int_on_threshold_mode_set
i3g4250d_int_x_threshold_set
i3g4250d_int_y_threshold_set
i3g4250d_int_z_threshold_set
i3g4250d_lp_bandwidth_set
i3g4250d_pin_int1_route_set
i3g4250d_pin_int2_route_set
i3g4250d_pin_mode_set
i3g4250d_pin_polarity_set
i3g4250d_self_test_set
i3g4250d_spi_mode_set

4.1.6 I3G4250D_Sensitivity

These functions convert raw-data into engineering units.

Collaboration diagram for I3G4250D_Sensitivity:

I3G4250D
I3G4250D_Sensitivity

Functions

- float_t i3g4250d_from_fs245dps_to_mdps (int16_t lsb)
- float_t i3g4250d_from_lsb_to_celsius (int16_t lsb)

4.1.6.1 Detailed Description

These functions convert raw-data into engineering units.

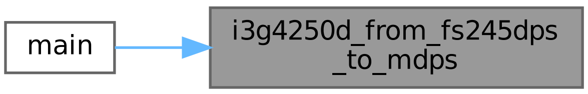
4.1.6.2 Function Documentation

4.1.6.2.1 i3g4250d_from_fs245dps_to_mdps()

```
float_t i3g4250d_from_fs245dps_to_mdps (
            int16_t lsb )
```

Referenced by main().

Here is the caller graph for this function:



4.1.6.2.2 i3g4250d_from_lsb_to_celsius()

```
float_t i3g4250d_from_lsb_to_celsius (
            int16_t lsb )
```

4.1.7 I3G4250D_data_generation

This section groups all the functions concerning data generation.

Collaboration diagram for I3G4250D_data_generation:

Functions

- int32_t i3g4250d_data_rate_get (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t ∗val)

  Accelerometer data rate selection.[get].
- int32_t i3g4250d_data_rate_set (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t val)

  Accelerometer data rate selection.[set].
- int32_t i3g4250d_flag_data_ready_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Accelerometer new data available.[get].
- int32_t i3g4250d_full_scale_get (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t ∗val)

  Gyroscope full-scale selection.[get].
- int32_t i3g4250d_full_scale_set (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t val)

  Gyroscope full-scale selection.[set].
- int32_t i3g4250d_status_reg_get (const stmdev_ctx_t ∗ctx, i3g4250d_status_reg_t ∗val)

  The STATUS_REG register is read by the primary interface.[get].

4.1.7.1 Detailed Description

This section groups all the functions concerning data generation.

4.1.7.2 Function Documentation

4.1.7.2.1 i3g4250d_data_rate_get()

```
int32_t i3g4250d_data_rate_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_dr_t * val )
```

Accelerometer data rate selection.[get].

Parameters

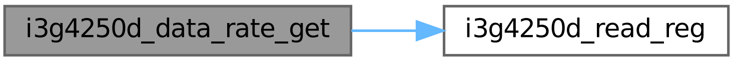
ctx	Read / write interface definitions.(ptr)
val	Get the values of dr in reg CTRL_REG1.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::dr, I3G4250D_CTRL_REG1, I3G4250D_ODR_100Hz, I3G4250D_ODR_200Hz, I3G4250D_ODR_400Hz, I3G4250D_ODR_800Hz, I3G4250D_ODR_OFF, I3G4250D_ODR_SLEEP, i3g4250d_read_reg(), and i3g4250d_ctrl_reg1_t::pd.

Here is the call graph for this function:



4.1.7.2.2 i3g4250d_data_rate_set()

```
int32_t i3g4250d_data_rate_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_dr_t val )
```

Accelerometer data rate selection.[set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of dr in reg CTRL_REG1

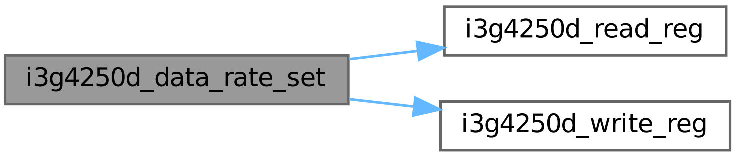
Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::dr, I3G4250D_CTRL_REG1, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg1_t::pd.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

main
i3g4250d_data_rate_set

4.1.7.2.3 i3g4250d_flag_data_ready_get()

```
int32_t i3g4250d_flag_data_ready_get (
              const stmdev_ctx_t * ctx,
              uint8_t * val )
```

Accelerometer new data available.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "zyxda" in reg STATUS_REG.(ptr)

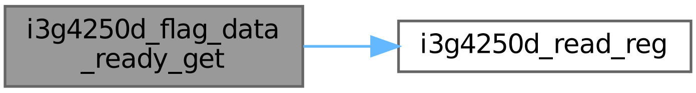
Return values

Interface	status (MANDATORY: return 0 -> no Error)

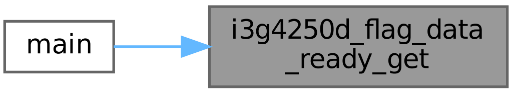
References i3g4250d_read_reg(), I3G4250D_STATUS_REG, and i3g4250d_status_reg_t::zyxda.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.7.2.4 i3g4250d_full_scale_get()

```
int32_t i3g4250d_full_scale_get (
              const stmdev_ctx_t * ctx,
              i3g4250d_fs_t * val )
```

Gyroscope full-scale selection.[get].

Parameters

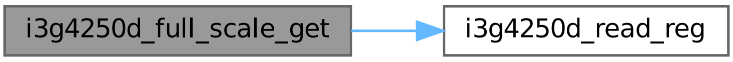
ctx	read / write interface definitions(ptr)
val	Get the values of fs in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::fs, I3G4250D_2000dps, I3G4250D_245dps, I3G4250D_500dps, I3G4250D_CTRL_REG4, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.7.2.5 i3g4250d_full_scale_set()

```
int32_t i3g4250d_full_scale_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_fs_t val )
```

Gyroscope full-scale selection.[set].

Parameters

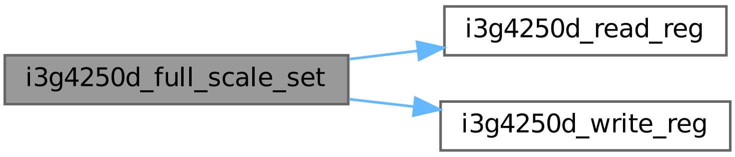
ctx	read / write interface definitions(ptr)
val	change the values of fs in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::fs, I3G4250D_CTRL_REG4, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.7.2.6 i3g4250d_status_reg_get()

```
int32_t i3g4250d_status_reg_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_status_reg_t * val )
```

The STATUS_REG register is read by the primary interface.[get].

Parameters

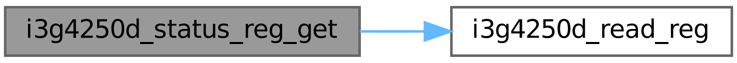
ctx	Read / write interface definitions.(ptr)
val	registers STATUS_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), and I3G4250D_STATUS_REG.

Here is the call graph for this function:



4.1.8 I3G4250D_Dataoutput

This section groups all the data output functions.

Collaboration diagram for I3G4250D_Dataoutput:

I3G4250D
I3G4250D_Dataoutput

Functions

- int32_t i3g4250d_angular_rate_raw_get (const stmdev_ctx_t ∗ctx, int16_t ∗val)

  Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].
- int32_t i3g4250d_temperature_raw_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Temperature data.[get].

4.1.8.1 Detailed Description

This section groups all the data output functions.

4.1.8.2 Function Documentation

4.1.8.2.1 i3g4250d_angular_rate_raw_get()

```
int32_t i3g4250d_angular_rate_raw_get (
            const stmdev_ctx_t * ctx,
            int16_t * val )
```

Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

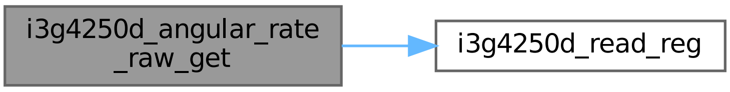
Return values

Interface	status (MANDATORY: return 0 -> no Error)

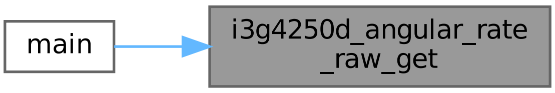
References I3G4250D_OUT_X_L, and i3g4250d_read_reg().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.8.2.2 i3g4250d_temperature_raw_get()

```
int32_t i3g4250d_temperature_raw_get (
            const stmdev_ctx_t * ctx,
            uint8_t * buff )
```

Temperature data.[get].

Parameters

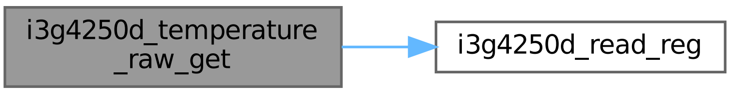
ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_OUT_TEMP, and i3g4250d_read_reg().

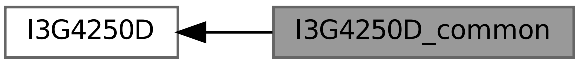
Here is the call graph for this function:



4.1.9 I3G4250D_common

This section groups common useful functions.

Collaboration diagram for I3G4250D_common:



Functions

- int32_t i3g4250d_boot_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reboot memory content. Reload the calibration parameters.[get].
- int32_t i3g4250d_boot_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reboot memory content. Reload the calibration parameters.[set].
- int32_t i3g4250d_data_format_get (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t ∗val)

  Big/Little Endian data selection.[get].
- int32_t i3g4250d_data_format_set (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t val)

  Big/Little Endian data selection.[set].
- int32_t i3g4250d_device_id_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Device Who amI.[get].
- int32_t i3g4250d_self_test_get (const stmdev_ctx_t ∗ctx, i3g4250d_st_t ∗val)

  Angular rate sensor self-test enable. [get].
- int32_t i3g4250d_self_test_set (const stmdev_ctx_t ∗ctx, i3g4250d_st_t val)

  Angular rate sensor self-test enable. [set].

4.1.9.1 Detailed Description

This section groups common useful functions.

4.1.9.2 Function Documentation

4.1.9.2.1 i3g4250d_boot_get()

```
int32_t i3g4250d_boot_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Reboot memory content. Reload the calibration parameters.[get].

Parameters

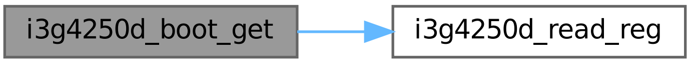
ctx	Read / write interface definitions.(ptr)
val	Get the values of boot in reg CTRL_REG5.(ptr)

Return values

interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::boot, I3G4250D_CTRL_REG5, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.9.2.2 i3g4250d_boot_set()

```
int32_t i3g4250d_boot_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Reboot memory content. Reload the calibration parameters.[set].

Parameters

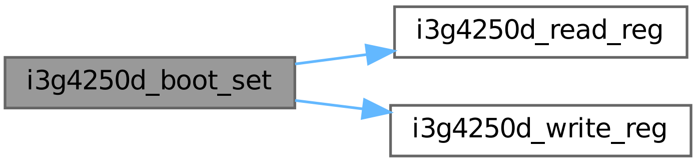
ctx	Read / write interface definitions.(ptr)
val	Change the values of boot in reg CTRL_REG5.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::boot, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.9.2.3 i3g4250d_data_format_get()

```
int32_t i3g4250d_data_format_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_ble_t * val )
```

Big/Little Endian data selection.[get].

Parameters

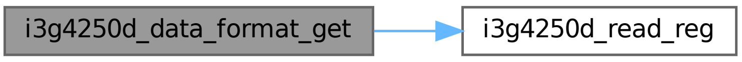
ctx	Read / write interface definitions.(ptr)
val	Get the values of "ble" in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::ble, I3G4250D_AUX_LSB_AT_LOW_ADD, I3G4250D_AUX_MSB_AT_LOW_ADD, I3G4250D_CTRL_REG4, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.9.2.4 i3g4250d_data_format_set()

```
int32_t i3g4250d_data_format_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_ble_t val )
```

Big/Little Endian data selection.[set].

Parameters

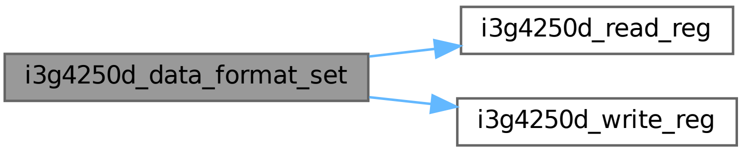
ctx	Read / write interface definitions.(ptr)
val	Change the values of "ble" in reg CTRL_REG4.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg4_t::ble, I3G4250D_CTRL_REG4, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.9.2.5 i3g4250d_device_id_get()

```
int32_t i3g4250d_device_id_get (
            const stmdev_ctx_t * ctx,
            uint8_t * buff )
```

Device Who amI.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
buff	Buffer that stores the data read.(ptr)

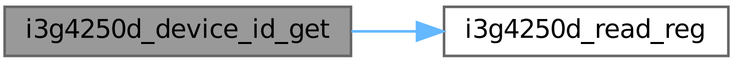
Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), and I3G4250D_WHO_AM_I.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:

main
i3g4250d_device_id_get

4.1.9.2.6 i3g4250d_self_test_get()

```
int32_t i3g4250d_self_test_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_st_t * val )
```

Angular rate sensor self-test enable. [get].

Parameters

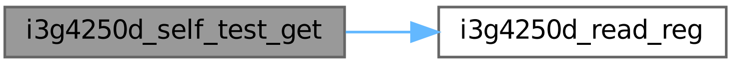
ctx	Read / write interface definitions.(ptr)
val	Get the values of st in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, I3G4250D_GY_ST_DISABLE, I3G4250D_GY_ST_NEGATIVE, I3G4250D_GY_ST_POSITIVE i3g4250d_read_reg(), and i3g4250d_ctrl_reg4_t::st.

Here is the call graph for this function:



4.1.9.2.7 i3g4250d_self_test_set()

```
int32_t i3g4250d_self_test_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_st_t val )
```

Angular rate sensor self-test enable. [set].

Parameters

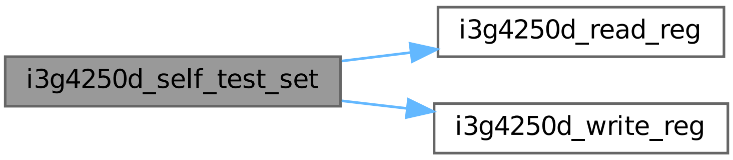
ctx	Read / write interface definitions.(ptr)
val	change the values of st in reg CTRL_REG4.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg4_t::st.

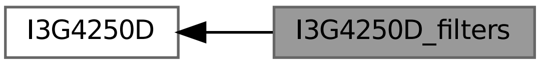
Here is the call graph for this function:



4.1.10 I3G4250D_filters

This section group all the functions concerning the filters configuration.

Collaboration diagram for I3G4250D_filters:



Functions

- int32_t i3g4250d_filter_path_get (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t ∗val)

  Out/FIFO selection path. [get].
- int32_t i3g4250d_filter_path_internal_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t ∗val)

  Interrupt generator selection path.[get].
- int32_t i3g4250d_filter_path_internal_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t val)

  Interrupt generator selection path.[set].
- int32_t i3g4250d_filter_path_set (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t val)

Out/FIFO selection path. [set].
- int32_t i3g4250d_hp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t ∗val)

  High-pass filter bandwidth selection.[get].
- int32_t i3g4250d_hp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t val)

  High-pass filter bandwidth selection.[set].
- int32_t i3g4250d_hp_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t ∗val)

  High-pass filter mode selection. [get].
- int32_t i3g4250d_hp_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t val)

  High-pass filter mode selection. [set].
- int32_t i3g4250d_hp_reference_value_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reference value for high-pass filter.[get].
- int32_t i3g4250d_hp_reference_value_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reference value for high-pass filter.[set].
- int32_t i3g4250d_lp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t ∗val)

  Lowpass filter bandwidth selection.[get].
- int32_t i3g4250d_lp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t val)

  Lowpass filter bandwidth selection.[set].

4.1.10.1 Detailed Description

This section group all the functions concerning the filters configuration.

4.1.10.2 Function Documentation

4.1.10.2.1 i3g4250d_filter_path_get()

```
int32_t i3g4250d_filter_path_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_out_sel_t * val )
```

Out/FIFO selection path. [get].

Parameters

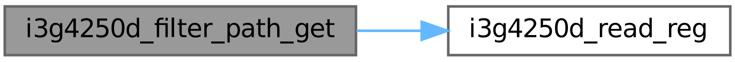
ctx	Read / write interface definitions.(ptr)
val	Get the values of out_sel in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, I3G4250D_LPF1_HP_LPF2_ON_OUT, I3G4250D_LPF1_HP_ON_OUT, I3G4250D_LPF1_LPF2_ON_OUT, I3G4250D_ONLY_LPF1_ON_OUT, i3g4250d_read_reg(), and i3g4250d_ctrl_reg5_t::out_sel.

Here is the call graph for this function:



4.1.10.2.2 i3g4250d_filter_path_internal_get()

```
int32_t i3g4250d_filter_path_internal_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_sel_t * val )
```

Interrupt generator selection path.[get].

Parameters

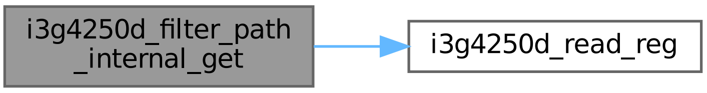
ctx	Read / write interface definitions.(ptr)
val	Get the values of int1_sel in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, I3G4250D_LPF1_HP_LPF2_ON_INT, I3G4250D_LPF1_HP_ON_INT, I3G4250D_LPF1_LPF2_ON_INT, I3G4250D_ONLY_LPF1_ON_INT, i3g4250d_read_reg(), and i3g4250d_ctrl_reg5_t::int1_sel.

Here is the call graph for this function:



4.1.10.2.3 i3g4250d_filter_path_internal_set()

```
int32_t i3g4250d_filter_path_internal_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_sel_t val )
```

Interrupt generator selection path.[set].

Parameters

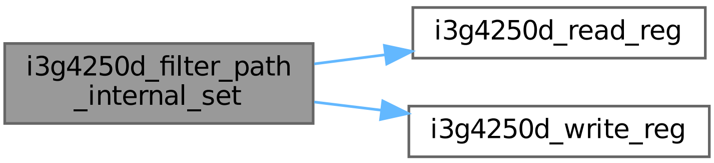
ctx	Read / write interface definitions.(ptr)
val	Change the values of int1_sel in reg CTRL_REG5

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg5_t::int1_sel.

Here is the call graph for this function:



4.1.10.2.4 i3g4250d_filter_path_set()

```
int32_t i3g4250d_filter_path_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_out_sel_t val )
```

Out/FIFO selection path. [set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "out_sel" in reg CTRL_REG5.

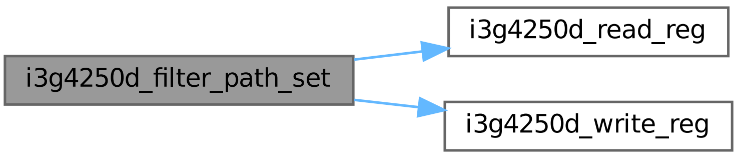
Return values

Interface	status (MANDATORY: return 0 -> no Error)

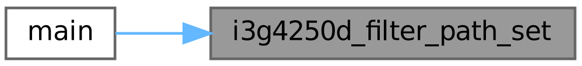
References i3g4250d_ctrl_reg5_t::hpen, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg5_t::out_sel.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.10.2.5 i3g4250d_hp_bandwidth_get()

```
int32_t i3g4250d_hp_bandwidth_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpcf_t * val )
```

High-pass filter bandwidth selection.[get].

Parameters

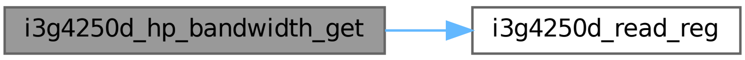
ctx	Read / write interface definitions.(ptr)
val	Get the values of hpcf in reg CTRL_REG2.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpcf, I3G4250D_CTRL_REG2, I3G4250D_HP_LEVEL_0, I3G4250D_HP_LEVEL_1, I3G4250D_HP_LEVEL_2, I3G4250D_HP_LEVEL_3, I3G4250D_HP_LEVEL_4, I3G4250D_HP_LEVEL_5, I3G4250D_HP_LEVEL_6, I3G4250D_HP_LEVEL_7, I3G4250D_HP_LEVEL_8, I3G4250D_HP_LEVEL_9, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.10.2.6 i3g4250d_hp_bandwidth_set()

```
int32_t i3g4250d_hp_bandwidth_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpcf_t val )
```

High-pass filter bandwidth selection.[set].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Change the values of "hpcf" in reg CTRL_REG2.

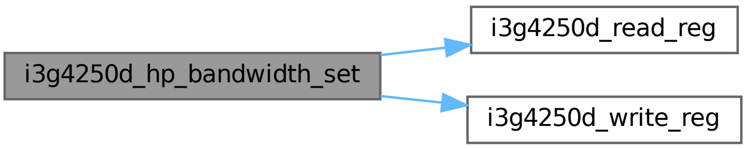
Return values

Interface	status (MANDATORY: return 0 -> no Error)

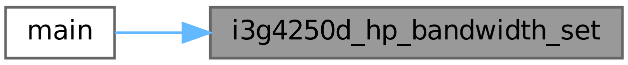
References i3g4250d_ctrl_reg2_t::hpcf, I3G4250D_CTRL_REG2, i3g4250d_read_reg(), and i3g4250d_write_reg().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.10.2.7 i3g4250d_hp_mode_get()

```
int32_t i3g4250d_hp_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpm_t * val )
```

High-pass filter mode selection. [get].

Parameters

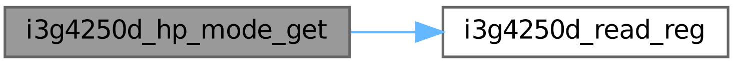
ctx	Read / write interface definitions.(ptr)
val	Get the values of hpm in reg CTRL_REG2.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpm, I3G4250D_CTRL_REG2, I3G4250D_HP_AUTO_RESET_ON_INT, I3G4250D_HP_NORMAL_MODE, I3G4250D_HP_NORMAL_MODE_WITH_RST, I3G4250D_HP_REFERENCE_SIGNAL, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.10.2.8 i3g4250d_hp_mode_set()

```
int32_t i3g4250d_hp_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_hpm_t val )
```

High-pass filter mode selection. [set].

Parameters

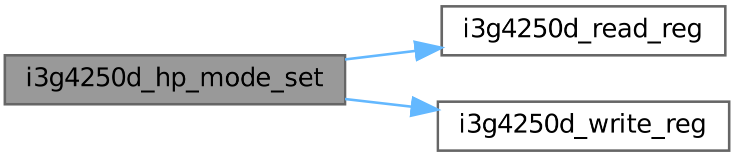
ctx	Read / write interface definitions.(ptr)
val	Change the values of "hpm" in reg CTRL_REG2.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg2_t::hpm, I3G4250D_CTRL_REG2, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.10.2.9 i3g4250d_hp_reference_value_get()

```
int32_t i3g4250d_hp_reference_value_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Reference value for high-pass filter.[get].

Parameters

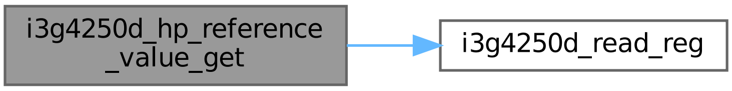
ctx	Read / write interface definitions.(ptr)
val	Get the values of ref in reg REFERENCE.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), I3G4250D_REFERENCE, and i3g4250d_reference_t::ref.

Here is the call graph for this function:



4.1.10.2.10 i3g4250d_hp_reference_value_set()

```
int32_t i3g4250d_hp_reference_value_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Reference value for high-pass filter.[set].

Parameters

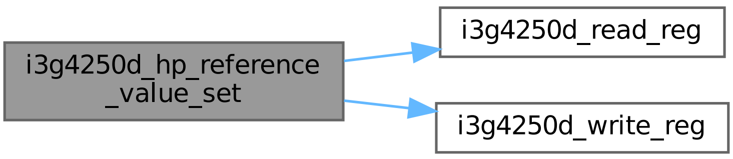
ctx	Read / write interface definitions.(ptr)
val	Change the values of ref in reg REFERENCE

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_read_reg(), I3G4250D_REFERENCE, i3g4250d_write_reg(), and i3g4250d_reference_t::ref.

Here is the call graph for this function:



4.1.10.2.11 i3g4250d_lp_bandwidth_get()

```
int32_t i3g4250d_lp_bandwidth_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_bw_t * val )
```

Lowpass filter bandwidth selection.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of "bw" in reg CTRL_REG1.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::bw, I3G4250D_CTRL_REG1, I3G4250D_CUT_OFF_HIGH, I3G4250D_CUT_OFF_LOW, I3G4250D_CUT_OFF_MEDIUM, I3G4250D_CUT_OFF_VERY_HIGH, and i3g4250d_read_reg().

Here is the call graph for this function:

4.1.10.2.12 i3g4250d_lp_bandwidth_set()

```
int32_t i3g4250d_lp_bandwidth_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_bw_t val )
```

Lowpass filter bandwidth selection.[set].

Parameters

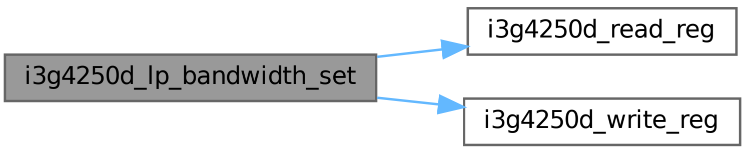
ctx	Read / write interface definitions.(ptr)
val	Change the values of "bw" in reg CTRL_REG1.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg1_t::bw, I3G4250D_CTRL_REG1, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.11 I3G4250D_serial_interface

This section groups all the functions concerning main serial interface management.

Collaboration diagram for I3G4250D_serial_interface:

I3G4250D
I3G4250D_serial_interface

Functions

- int32_t i3g4250d_spi_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t ∗val)

  SPI Serial Interface Mode selection.[get].
- int32_t i3g4250d_spi_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t val)

  SPI Serial Interface Mode selection.[set].

4.1.11.1 Detailed Description

This section groups all the functions concerning main serial interface management.

4.1.11.2 Function Documentation

4.1.11.2.1 i3g4250d_spi_mode_get()

```
int32_t i3g4250d_spi_mode_get (
            const stmdev_ctx_t ∗ ctx,
            i3g4250d_sim_t ∗ val )
```

SPI Serial Interface Mode selection.[get].

Parameters

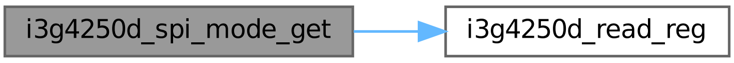
ctx	Read / write interface definitions.(ptr)
val	Get the values of sim in reg CTRL_REG4.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), I3G4250D_SPI_3_WIRE, I3G4250D_SPI_4_WIRE, and i3g4250d_ctrl_reg4_t::sim.

Here is the call graph for this function:



4.1.11.2.2 i3g4250d_spi_mode_set()

```
int32_t i3g4250d_spi_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_sim_t val )
```

SPI Serial Interface Mode selection.[set].

Parameters

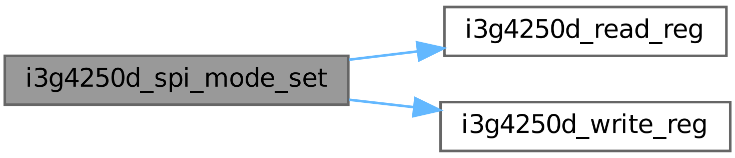
ctx	Read / write interface definitions.(ptr)
val	Change the values of sim in reg CTRL_REG4

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG4, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg4_t::sim.

Here is the call graph for this function:



4.1.12 I3G4250D_interrupt_pins

This section groups all the functions that manage interrupt pins.

Collaboration diagram for I3G4250D_interrupt_pins:

I3G4250D
I3G4250D_interrupt_pins

Functions

- int32_t i3g4250d_int_notification_get (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t ∗val)

  Latched/pulsed interrupt.[get].
- int32_t i3g4250d_int_notification_set (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t val)

  Latched/pulsed interrupt.[set].
- int32_t i3g4250d_pin_int1_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t ∗val)

  Select the signal that need to route on int1 pad.[get].
- int32_t i3g4250d_pin_int1_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t val)

  Select the signal that need to route on int1 pad.[set].
- int32_t i3g4250d_pin_int2_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t ∗val)

  Select the signal that need to route on int2 pad.[get].
- int32_t i3g4250d_pin_int2_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t val)

  Select the signal that need to route on int2 pad.[set].
- int32_t i3g4250d_pin_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t ∗val)

  Push-pull/open drain selection on interrupt pads.[get].
- int32_t i3g4250d_pin_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t val)

  Push-pull/open drain selection on interrupt pads.[set].
- int32_t i3g4250d_pin_polarity_get (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t ∗val)

  Pin active-high/low.[get].
- int32_t i3g4250d_pin_polarity_set (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t val)

  Pin active-high/low.[set].

4.1.12.1 Detailed Description

This section groups all the functions that manage interrupt pins.

4.1.12.2 Function Documentation

4.1.12.2.1 i3g4250d_int_notification_get()

```
int32_t i3g4250d_int_notification_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_lir_t * val )
```

Latched/pulsed interrupt.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of lir in reg INT1_CFG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, I3G4250D_INT_LATCHED, I3G4250D_INT_PULSED, i3g4250d_read_reg(), and i3g4250d_int1_cfg_t::lir.

Here is the call graph for this function:

i3g4250d_int_notification_get
i3g4250d_read_reg

4.1.12.2.2 i3g4250d_int_notification_set()

```
int32_t i3g4250d_int_notification_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_lir_t val )
```

Latched/pulsed interrupt.[set].

Parameters

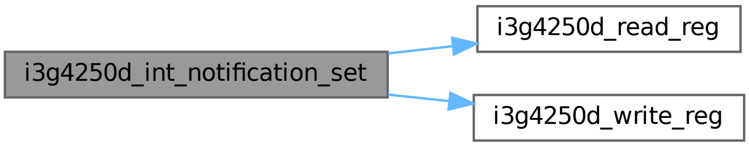
ctx	Read / write interface definitions.(ptr)
val	Change the values of lir in reg INT1_CFG.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_cfg_t::lir.

Here is the call graph for this function:



4.1.12.2.3 i3g4250d_pin_int1_route_get()

```
int32_t i3g4250d_pin_int1_route_get (
              const stmdev_ctx_t * ctx,
              i3g4250d_int1_route_t * val )
```

Select the signal that need to route on int1 pad.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Read CTRL_REG3 int1 pad.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i1_boot, i3g4250d_int1_route_t::i1_boot, i3g4250d_ctrl_reg3_t::i1_int1, i3g4250d_int1_route_t::i1_int1, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:

i3g4250d_pin_int1_route_get
i3g4250d_read_reg

4.1.12.2.4 i3g4250d_pin_int1_route_set()

```
int32_t i3g4250d_pin_int1_route_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_route_t val )
```

Select the signal that need to route on int1 pad.[set].

Parameters

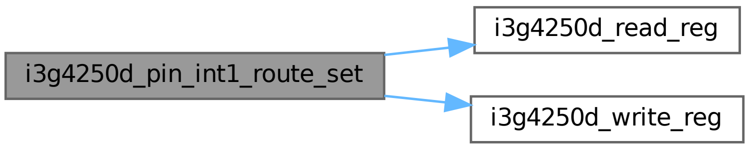
ctx	Read / write interface definitions.(ptr)
val	Configure CTRL_REG3 int1 pad

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i1_boot, i3g4250d_int1_route_t::i1_boot, i3g4250d_ctrl_reg3_t::i1_int1, i3g4250d_int1_route_t::i1_int1, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.12.2.5 i3g4250d_pin_int2_route_get()

```
int32_t i3g4250d_pin_int2_route_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int2_route_t * val )
```

Select the signal that need to route on int2 pad.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Read CTRL_REG3 int2 pad.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i2_drdy, i3g4250d_int2_route_t::i2_drdy, i3g4250d_ctrl_reg3_t::i2_empty, i3g4250d_int2_route_t::i2_empty, i3g4250d_ctrl_reg3_t::i2_orun, i3g4250d_int2_route_t::i2_orun, i3g4250d_ctrl_reg3_t::i2_wtm, i3g4250d_int2_route_t::i2_wtm, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:

i3g4250d_pin_int2_route_get
i3g4250d_read_reg

4.1.12.2.6 i3g4250d_pin_int2_route_set()

```
int32_t i3g4250d_pin_int2_route_set (
                const stmdev_ctx_t * ctx,
                i3g4250d_int2_route_t val )
```

Select the signal that need to route on int2 pad.[set].

Parameters

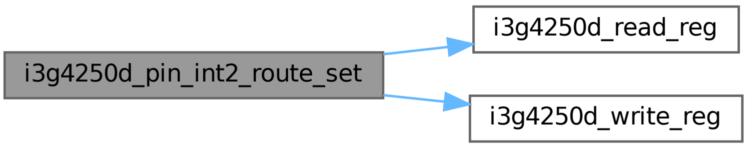
ctx	Read / write interface definitions.(ptr)
val	Configure CTRL_REG3 int2 pad

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::i2_drdy, i3g4250d_int2_route_t::i2_drdy, i3g4250d_ctrl_reg3_t::i2_empty, i3g4250d_int2_route_t::i2_empty, i3g4250d_ctrl_reg3_t::i2_orun, i3g4250d_int2_route_t::i2_orun, i3g4250d_ctrl_reg3_t::i2_wtm, i3g4250d_int2_route_t::i2_wtm, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.12.2.7 i3g4250d_pin_mode_get()

```
int32_t i3g4250d_pin_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_pp_od_t * val )
```

Push-pull/open drain selection on interrupt pads.[get].

Parameters

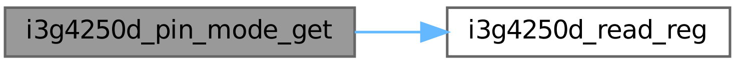
ctx	Read / write interface definitions.(ptr)
val	Get the values of pp_od in reg CTRL_REG3.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG3, I3G4250D_OPEN_DRAIN, I3G4250D_PUSH_PULL, i3g4250d_read_reg(), and i3g4250d_ctrl_reg3_t::pp_od.

Here is the call graph for this function:



4.1.12.2.8 i3g4250d_pin_mode_set()

```
int32_t i3g4250d_pin_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_pp_od_t val )
```

Push-pull/open drain selection on interrupt pads.[set].

Parameters

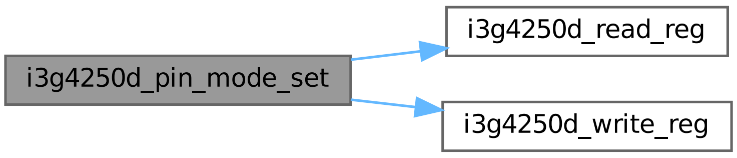
ctx	Read / write interface definitions.(ptr)
val	Change the values of pp_od in reg CTRL_REG3

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_CTRL_REG3, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_ctrl_reg3_t::pp_od.

Here is the call graph for this function:



4.1.12.2.9 i3g4250d_pin_polarity_get()

```
int32_t i3g4250d_pin_polarity_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_h_lactive_t * val )
```

Pin active-high/low.[get].

Parameters

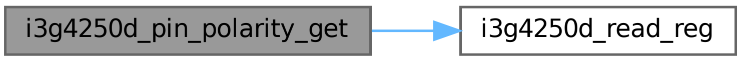
ctx	Read / write interface definitions.(ptr)
val	Get the values of h_lactive in reg CTRL_REG3.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::h_lactive, I3G4250D_ACTIVE_HIGH, I3G4250D_ACTIVE_LOW, I3G4250D_CTRL_REG3, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.12.2.10 i3g4250d_pin_polarity_set()

```
int32_t i3g4250d_pin_polarity_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_h_lactive_t val )
```

Pin active-high/low.[set].

Parameters

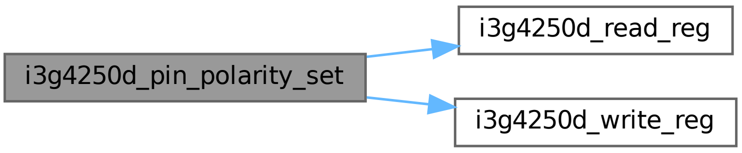
ctx	Read / write interface definitions.(ptr)
val	Change the values of h_lactive in reg CTRL_REG3.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg3_t::h_lactive, I3G4250D_CTRL_REG3, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.13 interrupt_on_threshold

This section groups all the functions that manage the event generation on threshold.

Collaboration diagram for interrupt_on_threshold:

I3G4250D
interrupt_on_threshold

Functions

- int32_t i3g4250d_int_on_threshold_conf_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[get].
- int32_t i3g4250d_int_on_threshold_conf_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[set].
- int32_t i3g4250d_int_on_threshold_dur_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Durationvalue.[get].
- int32_t i3g4250d_int_on_threshold_dur_set (const stmdev_ctx_t ∗ctx, uint8_t val)

Durationvalue.[set].

- int32_t i3g4250d_int_on_threshold_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t ∗val)

  AND/OR combination of interrupt events.[get].
- int32_t i3g4250d_int_on_threshold_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t val)

  AND/OR combination of interrupt events.[set].
- int32_t i3g4250d_int_on_threshold_src_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_src_t ∗val)

  int_on_threshold_src: [get]
- int32_t i3g4250d_int_x_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on X.[get].
- int32_t i3g4250d_int_x_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on X.[set].
- int32_t i3g4250d_int_y_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Y.[get].
- int32_t i3g4250d_int_y_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Y.[set].
- int32_t i3g4250d_int_z_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Z.[get].
- int32_t i3g4250d_int_z_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Z.[set].

4.1.13.1 Detailed Description

This section groups all the functions that manage the event generation on threshold.

4.1.13.2 Function Documentation

4.1.13.2.1 i3g4250d_int_on_threshold_conf_get()

```
int32_t i3g4250d_int_on_threshold_conf_get (
              const stmdev_ctx_t ∗ ctx,
              i3g4250d_int1_cfg_t ∗ val )
```

Configure the interrupt threshold sign.[get].

Parameters

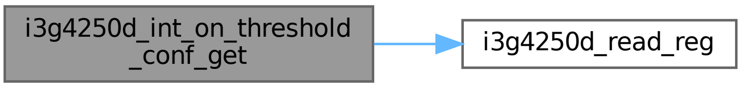
ctx	Read / write interface definitions.(ptr)
val	Struct of registers from INT1_CFG to.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.13.2.2 i3g4250d_int_on_threshold_conf_set()

```
int32_t i3g4250d_int_on_threshold_conf_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_cfg_t * val )
```

Configure the interrupt threshold sign.[set].

Parameters

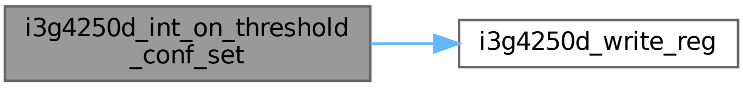
ctx	Read / write interface definitions.(ptr)
val	Struct of registers INT1_CFG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_CFG, and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.13.2.3 i3g4250d_int_on_threshold_dur_get()

```
int32_t i3g4250d_int_on_threshold_dur_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Durationvalue.[get].

Parameters

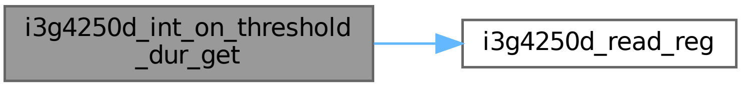
ctx	Read / write interface definitions.(ptr)
val	Get the values of d in reg INT1_DURATION.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_duration_t::d, I3G4250D_INT1_DURATION, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.13.2.4 i3g4250d_int_on_threshold_dur_set()

```
int32_t i3g4250d_int_on_threshold_dur_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

Durationvalue.[set].

Parameters

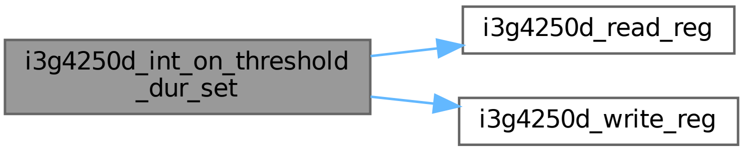
ctx	Read / write interface definitions.(ptr)
val	Change the values of d in reg INT1_DURATION

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_duration_t::d, I3G4250D_INT1_DURATION, i3g4250d_read_reg(), i3g4250d_write_reg(), PROPERTY_DISABLE, PROPERTY_ENABLE, and i3g4250d_int1_duration_t::wait.

Here is the call graph for this function:



4.1.13.2.5 i3g4250d_int_on_threshold_mode_get()

```
int32_t i3g4250d_int_on_threshold_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_and_or_t * val )
```

AND/OR combination of interrupt events.[get].

Parameters

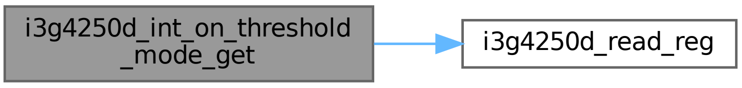
ctx	Read / write interface definitions.(ptr)
val	Get the values of and_or in reg INT1_CFG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_cfg_t::and_or, I3G4250D_INT1_CFG, I3G4250D_INT1_ON_TH_AND, I3G4250D_INT1_ON_TH_OR, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.13.2.6 i3g4250d_int_on_threshold_mode_set()

```
int32_t i3g4250d_int_on_threshold_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_and_or_t val )
```

AND/OR combination of interrupt events.[set].

Parameters

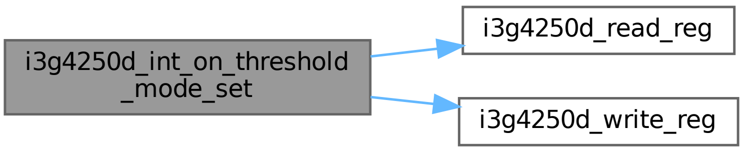
ctx	Read / write interface definitions.(ptr)
val	Change the values of and_or in reg INT1_CFG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_int1_cfg_t::and_or, I3G4250D_INT1_CFG, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.13.2.7 i3g4250d_int_on_threshold_src_get()

```
int32_t i3g4250d_int_on_threshold_src_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_int1_src_t * val )
```

int_on_threshold_src: [get]

Parameters

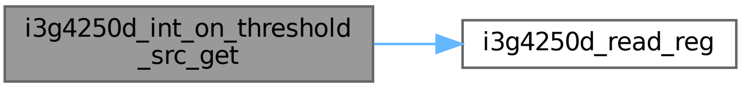
ctx	Read / write interface definitions.(ptr)
val	Union of registers from INT1_SRC to.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_SRC, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.13.2.8 i3g4250d_int_x_threshold_get()

```
int32_t i3g4250d_int_x_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on X.[get].

Parameters

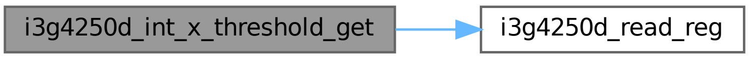
ctx	Read / write interface definitions.(ptr)
val	Get the values of thsx in reg INT1_TSH_XH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_XH, I3G4250D_INT1_TSH_XL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_xh_t::thsx.

Here is the call graph for this function:



4.1.13.2.9 i3g4250d_int_x_threshold_set()

```
int32_t i3g4250d_int_x_threshold_set (
            const stmdev_ctx_t * ctx,
            uint16_t val )
```

Interrupt threshold on X.[set].

Parameters

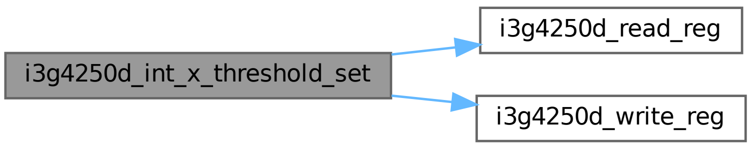
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsx in reg INT1_TSH_XH

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_XH, I3G4250D_INT1_TSH_XL, i3g4250d_read_reg(), i3g4250d_write_reg(), i3g4250d_int1_tsh_xh_t::thsx, and i3g4250d_int1_tsh_xl_t::thsx.

Here is the call graph for this function:



4.1.13.2.10 i3g4250d_int_y_threshold_get()

```
int32_t i3g4250d_int_y_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on Y.[get].

Parameters

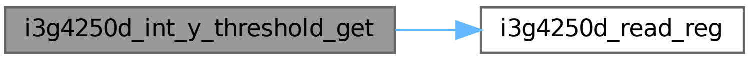
ctx	Read / write interface definitions.(ptr)
val	Get the values of thsy in reg INT1_TSH_YH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_YH, I3G4250D_INT1_TSH_YL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_yh_t::thsy.

Here is the call graph for this function:



4.1.13.2.11 i3g4250d_int_y_threshold_set()

```
int32_t i3g4250d_int_y_threshold_set (
            const stmdev_ctx_t * ctx,
            uint16_t val )
```

Interrupt threshold on Y.[set].

Parameters

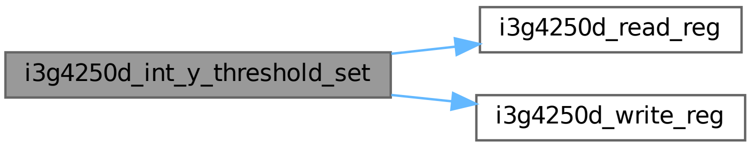
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsy in reg INT1_TSH_YH

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_YH, I3G4250D_INT1_TSH_YL, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_tsh_yh_t::thsy.

Here is the call graph for this function:



4.1.13.2.12 i3g4250d_int_z_threshold_get()

```
int32_t i3g4250d_int_z_threshold_get (
            const stmdev_ctx_t * ctx,
            uint16_t * val )
```

Interrupt threshold on Z.[get].

Parameters

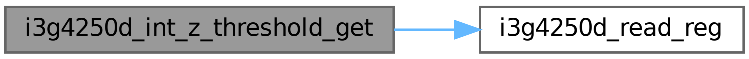
ctx	Read / write interface definitions.(ptr)
val	Get the values of thsz in reg INT1_TSH_ZH.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_ZH, I3G4250D_INT1_TSH_ZL, i3g4250d_read_reg(), and i3g4250d_int1_tsh_zh_t::thsz.

Here is the call graph for this function:



4.1.13.2.13 i3g4250d_int_z_threshold_set()

```
int32_t i3g4250d_int_z_threshold_set (
            const stmdev_ctx_t * ctx,
            uint16_t val )
```

Interrupt threshold on Z.[set].

Parameters

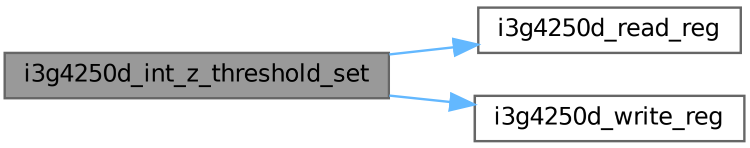
ctx	Read / write interface definitions.(ptr)
val	Change the values of thsz in reg INT1_TSH_ZH.

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_INT1_TSH_ZH, I3G4250D_INT1_TSH_ZL, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_int1_tsh_zh_t::thsz.

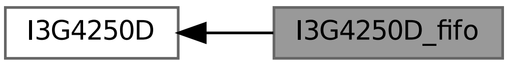
Here is the call graph for this function:



4.1.14 I3G4250D_fifo

This section group all the functions concerning the fifo usage.

Collaboration diagram for I3G4250D_fifo:



Functions

- int32_t i3g4250d_fifo_data_level_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO stored data level[get].
- int32_t i3g4250d_fifo_empty_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOemptybit.[get].
- int32_t i3g4250d_fifo_enable_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOenable.[get].
- int32_t i3g4250d_fifo_enable_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFOenable.[set].
- int32_t i3g4250d_fifo_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t ∗val)

  FIFO mode selection.[get].
- int32_t i3g4250d_fifo_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t val)

  FIFO mode selection.[set].
- int32_t i3g4250d_fifo_ovr_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Overrun bit status.[get].
- int32_t i3g4250d_fifo_watermark_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO watermark level selection.[get].
- int32_t i3g4250d_fifo_watermark_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFO watermark level selection.[set].
- int32_t i3g4250d_fifo_wtm_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)

4.1.14.1 Detailed Description

This section group all the functions concerning the fifo usage.

4.1.14.2 Function Documentation

4.1.14.2.1 i3g4250d_fifo_data_level_get()

```
int32_t i3g4250d_fifo_data_level_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFO stored data level[get].

Parameters

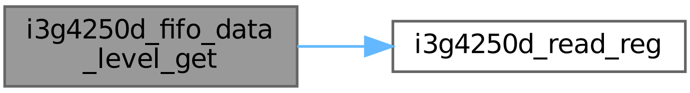
ctx	Read / write interface definitions.(ptr)
val	Get the values of fss in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_src_reg_t::fss, I3G4250D_FIFO_SRC_REG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.14.2.2 i3g4250d_fifo_empty_flag_get()

```
int32_t i3g4250d_fifo_empty_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFOemptybit.[get].

Parameters

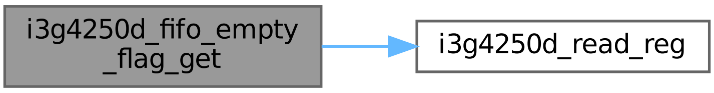
ctx	Read / write interface definitions.(ptr)
val	Get the values of empty in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_src_reg_t::empty, I3G4250D_FIFO_SRC_REG, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.14.2.3 i3g4250d_fifo_enable_get()

```
int32_t i3g4250d_fifo_enable_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFOenable.[get].

Parameters

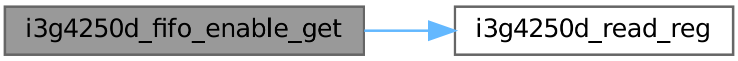
ctx	Read / write interface definitions.(ptr)
val	Get the values of fifo_en in reg CTRL_REG5.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::fifo_en, I3G4250D_CTRL_REG5, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.14.2.4 i3g4250d_fifo_enable_set()

```
int32_t i3g4250d_fifo_enable_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

FIFOenable.[set].

Parameters

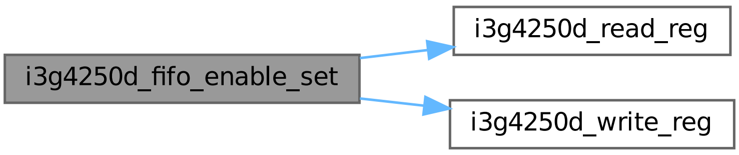
ctx	Read / write interface definitions.(ptr)
val	Change the values of fifo_en in reg CTRL_REG5

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_ctrl_reg5_t::fifo_en, I3G4250D_CTRL_REG5, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.14.2.5 i3g4250d_fifo_mode_get()

```
int32_t i3g4250d_fifo_mode_get (
            const stmdev_ctx_t * ctx,
            i3g4250d_fifo_mode_t * val )
```

FIFO mode selection.[get].

Parameters

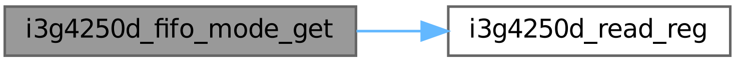
ctx	Read / write interface definitions.(ptr)
val	Get the values of fm in reg FIFO_CTRL_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_ctrl_reg_t::fm, I3G4250D_FIFO_BYPASS_MODE, I3G4250D_FIFO_CTRL_REG, I3G4250D_FIFO_MODE, I3G4250D_FIFO_STREAM_MODE, and i3g4250d_read_reg().

Here is the call graph for this function:



4.1.14.2.6 i3g4250d_fifo_mode_set()

```
int32_t i3g4250d_fifo_mode_set (
            const stmdev_ctx_t * ctx,
            i3g4250d_fifo_mode_t val )
```

FIFO mode selection.[set].

Parameters

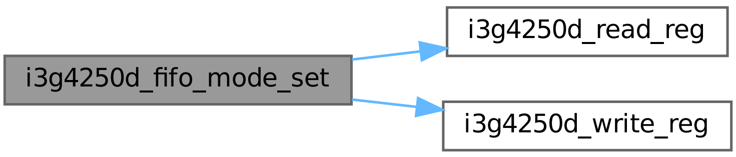
ctx	Read / write interface definitions.(ptr)
val	Change the values of fm in reg FIFO_CTRL_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References i3g4250d_fifo_ctrl_reg_t::fm, I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), and i3g4250d_write_reg().

Here is the call graph for this function:



4.1.14.2.7 i3g4250d_fifo_ovr_flag_get()

```
int32_t i3g4250d_fifo_ovr_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Overrun bit status.[get].

Parameters

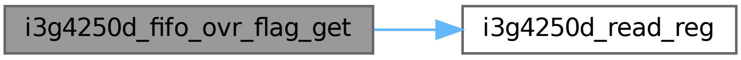
ctx	Read / write interface definitions.(ptr)
val	Get the values of ovrn in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_SRC_REG, i3g4250d_read_reg(), and i3g4250d_fifo_src_reg_t::ovrn.

Here is the call graph for this function:



4.1.14.2.8 i3g4250d_fifo_watermark_get()

```
int32_t i3g4250d_fifo_watermark_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

FIFO watermark level selection.[get].

Parameters

ctx	Read / write interface definitions.(ptr)
val	Get the values of wtm in reg FIFO_CTRL_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), and i3g4250d_fifo_ctrl_reg_t::wtm.

Here is the call graph for this function:

i3g4250d_fifo_watermark_get
i3g4250d_read_reg

4.1.14.2.9 i3g4250d_fifo_watermark_set()

```
int32_t i3g4250d_fifo_watermark_set (
            const stmdev_ctx_t * ctx,
            uint8_t val )
```

FIFO watermark level selection.[set].

Parameters

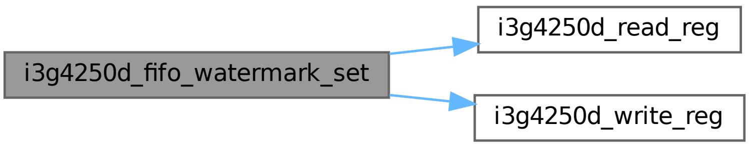
ctx	Read / write interface definitions.(ptr)
val	Change the values of wtm in reg FIFO_CTRL_REG

Return values

Interface	status (MANDATORY: return 0 -> no Error)

References I3G4250D_FIFO_CTRL_REG, i3g4250d_read_reg(), i3g4250d_write_reg(), and i3g4250d_fifo_ctrl_reg_t::wtm.

Here is the call graph for this function:



4.1.14.2.10 i3g4250d_fifo_wtm_flag_get()

```
int32_t i3g4250d_fifo_wtm_flag_get (
            const stmdev_ctx_t * ctx,
            uint8_t * val )
```

Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)

Parameters

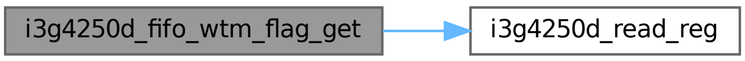
ctx	Read / write interface definitions.(ptr)
val	Get the values of wtm in reg FIFO_SRC_REG.(ptr)

Return values

Interface	status (MANDATORY: return 0 -> no Error)

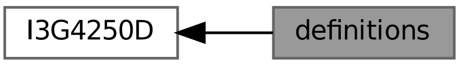
References I3G4250D_FIFO_SRC_REG, i3g4250d_read_reg(), and i3g4250d_fifo_src_reg_t::wtm.

Here is the call graph for this function:



4.1.15 definitions

Collaboration diagram for definitions:



Macros

- #define DRV_BIG_ENDIAN 4321
- #define DRV_BYTE_ORDER DRV_LITTLE_ENDIAN
- #define DRV_LITTLE_ENDIAN 1234

4.1.15.1 Detailed Description

4.1.15.2 Macro Definition Documentation

4.1.15.2.1 DRV_BIG_ENDIAN

```
#define DRV_BIG_ENDIAN 4321
```

4.1.15.2.2 DRV_BYTE_ORDER

```
#define DRV_BYTE_ORDER DRV_LITTLE_ENDIAN
```

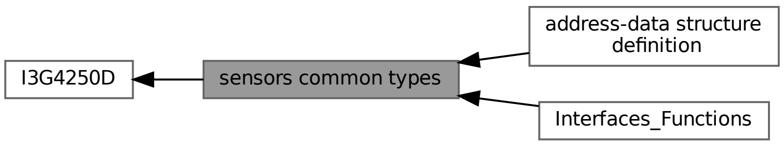
if DRV_BYTE_ORDER is not defined, choose the endianness of your architecture by uncommenting the define which fits your platform endianness

4.1.15.2.3 DRV_LITTLE_ENDIAN

```
#define DRV_LITTLE_ENDIAN 1234
```

4.1.16 sensors common types

Collaboration diagram for sensors common types:



Modules

- address-data structure definition

  This structure is useful to load a predefined configuration of a sensor. You can create a sensor configuration by your own or using Unico / Unicleo tools available on STMicroelectronics web site.
- Interfaces_Functions

  This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.

Data Structures

- struct bitwise_t

Macros

- #define MEMS_SHARED_TYPES
- #define MEMS_UCF_SHARED_TYPES
- #define PROPERTY_DISABLE (0U)
- #define PROPERTY_ENABLE (1U)

4.1.16.1 Detailed Description

4.1.16.2 Macro Definition Documentation

4.1.16.2.1 MEMS_SHARED_TYPES

```
#define MEMS_SHARED_TYPES
```

4.1.16.2.2 MEMS_UCF_SHARED_TYPES

```
#define MEMS_UCF_SHARED_TYPES
```

4.1.16.2.3 PROPERTY_DISABLE

```
#define PROPERTY_DISABLE (0U)
```

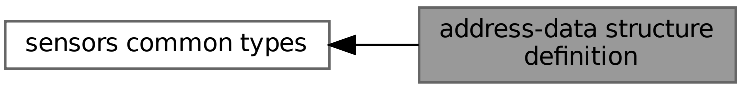
4.1.16.2.4 PROPERTY_ENABLE

```
#define PROPERTY_ENABLE (1U)
```

4.1.16.3 address-data structure definition

This structure is useful to load a predefined configuration of a sensor. You can create a sensor configuration by your own or using Unico / Unicleo tools available on STMicroelectronics web site.

Collaboration diagram for address-data structure definition:



Data Structures

- struct ucf_line_t

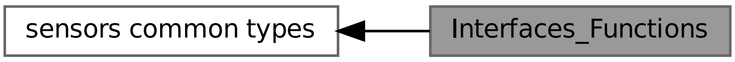
4.1.16.3.1 Detailed Description

This structure is useful to load a predefined configuration of a sensor. You can create a sensor configuration by your own or using Unico / Unicleo tools available on STMicroelectronics web site.

4.1.16.4 Interfaces_Functions

This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.

Collaboration diagram for Interfaces_Functions:



Data Structures

- struct stmdev_ctx_t

Typedefs

- typedef void(∗ stmdev_mdelay_ptr) (uint32_t millisec)
- typedef int32_t(∗ stmdev_read_ptr) (void ∗, uint8_t, uint8_t ∗, uint16_t)
- typedef int32_t(∗ stmdev_write_ptr) (void ∗, uint8_t, const uint8_t ∗, uint16_t)

4.1.16.4.1 Detailed Description

This section provide a set of functions used to read and write a generic register of the device. MANDATORY: return 0 -> no Error.

4.1.16.4.2 Typedef Documentation

4.1.16.4.2.1 stmdev_mdelay_ptr

```
typedef void(* stmdev_mdelay_ptr) (uint32_t millisec)
```

4.1.16.4.2.2 stmdev_read_ptr

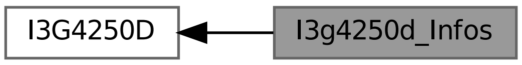
```
typedef int32_t(* stmdev_read_ptr) (void *, uint8_t, uint8_t *, uint16_t)
```

4.1.16.4.2.3 stmdev_write_ptr

```
typedef int32_t(* stmdev_write_ptr) (void *, uint8_t, const uint8_t *, uint16_t)
```

4.1.17 I3g4250d_Infos

Collaboration diagram for I3g4250d_Infos:



Macros

- #define I3G4250D_I2C_ADD_H 0xD3U
- #define I3G4250D_I2C_ADD_L 0xD1U
- #define I3G4250D_ID 0xD3U

4.1.17.1 Detailed Description

4.1.17.2 Macro Definition Documentation

4.1.17.2.1 I3G4250D_I2C_ADD_H

```
#define I3G4250D_I2C_ADD_H 0xD3U
```

4.1.17.2.2 I3G4250D_I2C_ADD_L

```
#define I3G4250D_I2C_ADD_L 0xD1U
```

I2C Device Address 8 bit format if SA0=0 -> 0xD1 if SA0=1 -> 0xD3

4.1.17.2.3 I3G4250D_ID

```
#define I3G4250D_ID 0xD3U
```

Device Identification (Who am I)

4.1.18 LSM9DS1_Register_Union

This union group all the registers having a bit-field description. This union is useful but it's not needed by the driver.

Collaboration diagram for LSM9DS1_Register_Union:

I3G4250D
LSM9DS1_Register_Union

Data Structures

- union i3g4250d_reg_t

4.1.18.1 Detailed Description

This union group all the registers having a bit-field description. This union is useful but it's not needed by the driver.

REMOVING this union you are compliant with: MISRA-C 2012 [Rule 19.2] -> " Union are not allowed "

4.2 CMSIS

Collaboration diagram for CMSIS:

CMSIS
Stm32f4xx_system

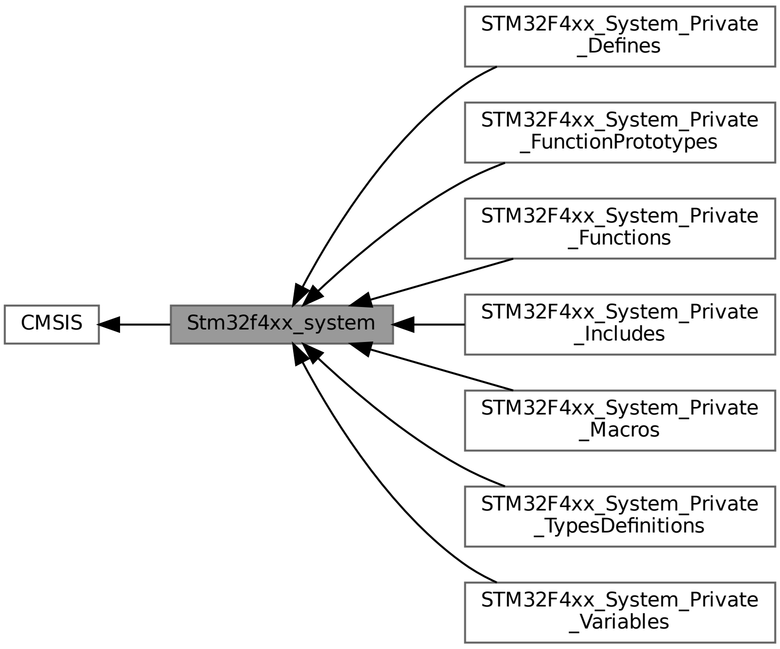
Modules

- Stm32f4xx_system

4.2.1 Detailed Description

4.2.2 Stm32f4xx_system

Collaboration diagram for Stm32f4xx_system:



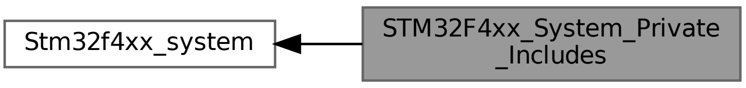
Modules

- STM32F4xx_System_Private_Includes
- STM32F4xx_System_Private_TypesDefinitions
- STM32F4xx_System_Private_Defines
- STM32F4xx_System_Private_Macros
- STM32F4xx_System_Private_Variables
- STM32F4xx_System_Private_FunctionPrototypes
- STM32F4xx_System_Private_Functions

4.2.2.1 Detailed Description

4.2.2.2 STM32F4xx_System_Private_Includes

Collaboration diagram for STM32F4xx_System_Private_Includes:



Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

4.2.2.2.1 Detailed Description

4.2.2.2.2 Macro Definition Documentation

4.2.2.2.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)25000000)
```

Default value of the External oscillator in Hz

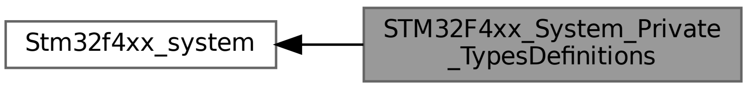
4.2.2.2.2.2 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000)
```

Value of the Internal oscillator in Hz

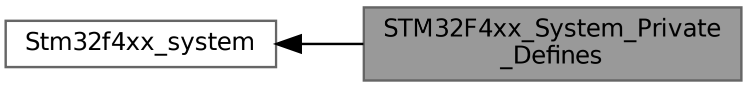
4.2.2.3 STM32F4xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F4xx_System_Private_TypesDefinitions:



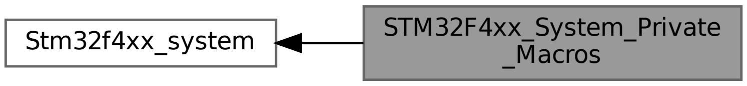
4.2.2.4 STM32F4xx_System_Private_Defines

Collaboration diagram for STM32F4xx_System_Private_Defines:



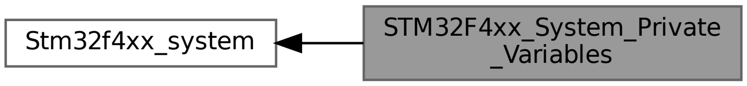
4.2.2.5 STM32F4xx_System_Private_Macros

Collaboration diagram for STM32F4xx_System_Private_Macros:



4.2.2.6 STM32F4xx_System_Private_Variables

Collaboration diagram for STM32F4xx_System_Private_Variables:



Variables

- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}
- uint32_t SystemCoreClock = 16000000

4.2.2.6.1 Detailed Description

4.2.2.6.2 Variable Documentation

4.2.2.6.2.1 AHBPrescTable

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

Referenced by SystemCoreClockUpdate().

4.2.2.6.2.2 APBPrescTable

```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

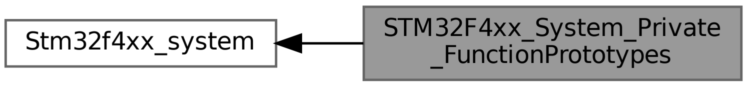
4.2.2.6.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

Referenced by SystemCoreClockUpdate().

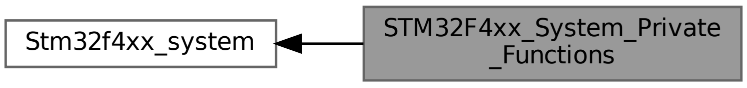
4.2.2.7 STM32F4xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F4xx_System_Private_FunctionPrototypes:



4.2.2.8 STM32F4xx_System_Private_Functions

Collaboration diagram for STM32F4xx_System_Private_Functions:



Functions

- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

4.2.2.8.1 Detailed Description

4.2.2.8.2 Function Documentation

4.2.2.8.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
            void  )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the HSI_VALUE(∗)
- If SYSCLK source is HSE, SystemCoreClock will contain the HSE_VALUE(∗∗)
- If SYSCLK source is PLL, SystemCoreClock will contain the HSE_VALUE(∗∗) or HSI_VALUE(∗) multiplied/divided by the PLL factors.

(∗) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file (default value 16 MHz) but the real value may vary depending on the variations in voltage and temperature.

(∗∗) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file (its value depends on the application requirements), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	

Return values

None	

< Value of the Internal oscillator in Hz

< Default value of the External oscillator in Hz

< Default value of the External oscillator in Hz

< Value of the Internal oscillator in Hz

< Value of the Internal oscillator in Hz

References AHBPrescTable, HSE_VALUE, HSI_VALUE, and SystemCoreClock.

4.2.2.8.2.2 SystemInit()

```
void SystemInit (
             void  )
```

Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Parameters

None	

Return values

None	

Chapter 5

Data Structure Documentation

5.1 bitwise_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t bit0: 1
- uint8_t bit1: 1
- uint8_t bit2: 1
- uint8_t bit3: 1
- uint8_t bit4: 1
- uint8_t bit5: 1
- uint8_t bit6: 1
- uint8_t bit7: 1

5.1.1 Field Documentation

5.1.1.1 bit0

```
uint8_t bitwise_t::bit0
```

5.1.1.2 bit1

```
uint8_t bitwise_t::bit1
```

5.1.1.3 bit2

```
uint8_t bitwise_t::bit2
```

5.1.1.4 bit3

```
uint8_t bitwise_t::bit3
```

5.1.1.5 bit4

```
uint8_t bitwise_t::bit4
```

5.1.1.6 bit5

```
uint8_t bitwise_t::bit5
```

5.1.1.7 bit6

```
uint8_t bitwise_t::bit6
```

5.1.1.8 bit7

```
uint8_t bitwise_t::bit7
```

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

- i3g4250d_reg.h

5.2 i3g4250d_ctrl_reg1_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t bw: 2
- uint8_t dr: 2
- uint8_t pd: 4

5.2.1 Field Documentation

5.2.1.1 bw

```
uint8_t i3g4250d_ctrl_reg1_t::bw
```

Referenced by i3g4250d_lp_bandwidth_get(), and i3g4250d_lp_bandwidth_set().

5.2.1.2 dr

```
uint8_t i3g4250d_ctrl_reg1_t::dr
```

Referenced by i3g4250d_data_rate_get(), and i3g4250d_data_rate_set().

5.2.1.3 pd

```
uint8_t i3g4250d_ctrl_reg1_t::pd
```

Referenced by i3g4250d_data_rate_get(), and i3g4250d_data_rate_set().

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

- i3g4250d_reg.h

5.3 i3g4250d_ctrl_reg2_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t hpcf: 4
- uint8_t hpm: 2
- uint8_t not_used_01: 2

5.3.1 Field Documentation

5.3.1.1 hpcf

```
uint8_t i3g4250d_ctrl_reg2_t::hpcf
```

Referenced by i3g4250d_hp_bandwidth_get(), and i3g4250d_hp_bandwidth_set().

5.3.1.2 hpm

```
uint8_t i3g4250d_ctrl_reg2_t::hpm
```

Referenced by i3g4250d_hp_mode_get(), and i3g4250d_hp_mode_set().

5.3.1.3 not_used_01

```
uint8_t i3g4250d_ctrl_reg2_t::not_used_01
```

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

- i3g4250d_reg.h

5.4 i3g4250d_ctrl_reg3_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t h_lactive: 1
- uint8_t i1_boot: 1
- uint8_t i1_int1: 1
- uint8_t i2_drdy: 1
- uint8_t i2_empty: 1
- uint8_t i2_orun: 1
- uint8_t i2_wtm: 1
- uint8_t pp_od: 1

5.4.1 Field Documentation

5.4.1.1 h_lactive

```
uint8_t i3g4250d_ctrl_reg3_t::h_lactive
```

Referenced by i3g4250d_pin_polarity_get(), and i3g4250d_pin_polarity_set().

5.4.1.2 i1_boot

```
uint8_t i3g4250d_ctrl_reg3_t::i1_boot
```

Referenced by i3g4250d_pin_int1_route_get(), and i3g4250d_pin_int1_route_set().

5.4.1.3 i1_int1

```
uint8_t i3g4250d_ctrl_reg3_t::i1_int1
```

Referenced by i3g4250d_pin_int1_route_get(), and i3g4250d_pin_int1_route_set().

5.4.1.4 i2_drdy

```
uint8_t i3g4250d_ctrl_reg3_t::i2_drdy
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.4.1.5 i2_empty

```
uint8_t i3g4250d_ctrl_reg3_t::i2_empty
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.4.1.6 i2_orun

```
uint8_t i3g4250d_ctrl_reg3_t::i2_orun
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.4.1.7 i2_wtm

```
uint8_t i3g4250d_ctrl_reg3_t::i2_wtm
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.4.1.8 pp_od

```
uint8_t i3g4250d_ctrl_reg3_t::pp_od
```

Referenced by i3g4250d_pin_mode_get(), and i3g4250d_pin_mode_set().

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

- i3g4250d_reg.h

5.5 i3g4250d_ctrl_reg4_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t ble: 1
- uint8_t fs: 2
- uint8_t not_used_01: 1
- uint8_t not_used_02: 1
- uint8_t sim: 1
- uint8_t st: 2

5.5.1 Field Documentation

5.5.1.1 ble

```
uint8_t i3g4250d_ctrl_reg4_t::ble
```

Referenced by i3g4250d_data_format_get(), and i3g4250d_data_format_set().

5.5.1.2 fs

```
uint8_t i3g4250d_ctrl_reg4_t::fs
```

Referenced by i3g4250d_full_scale_get(), and i3g4250d_full_scale_set().

5.5.1.3 not_used_01

```
uint8_t i3g4250d_ctrl_reg4_t::not_used_01
```

5.5.1.4 not_used_02

```
uint8_t i3g4250d_ctrl_reg4_t::not_used_02
```

5.5.1.5 sim

```
uint8_t i3g4250d_ctrl_reg4_t::sim
```

Referenced by i3g4250d_spi_mode_get(), and i3g4250d_spi_mode_set().

5.5.1.6 st

```
uint8_t i3g4250d_ctrl_reg4_t::st
```

Referenced by i3g4250d_self_test_get(), and i3g4250d_self_test_set().

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

- i3g4250d_reg.h

5.6 i3g4250d_ctrl_reg5_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t boot: 1
- uint8_t fifo_en: 1
- uint8_t hpen: 1
- uint8_t int1_sel: 2
- uint8_t not_used_01: 1
- uint8_t out_sel: 2

5.6.1 Field Documentation

5.6.1.1 boot

```
uint8_t i3g4250d_ctrl_reg5_t::boot
```

Referenced by i3g4250d_boot_get(), and i3g4250d_boot_set().

5.6.1.2 fifo_en

```
uint8_t i3g4250d_ctrl_reg5_t::fifo_en
```

Referenced by i3g4250d_fifo_enable_get(), and i3g4250d_fifo_enable_set().

5.6.1.3 hpen

```
uint8_t i3g4250d_ctrl_reg5_t::hpen
```

Referenced by i3g4250d_filter_path_get(), i3g4250d_filter_path_internal_get(), i3g4250d_filter_path_internal_set(), and i3g4250d_filter_path_set().

5.6.1.4 int1_sel

```
uint8_t i3g4250d_ctrl_reg5_t::int1_sel
```

Referenced by i3g4250d_filter_path_internal_get(), and i3g4250d_filter_path_internal_set().

5.6.1.5 not_used_01

```
uint8_t i3g4250d_ctrl_reg5_t::not_used_01
```

5.6.1.6 out_sel

```
uint8_t i3g4250d_ctrl_reg5_t::out_sel
```

Referenced by i3g4250d_filter_path_get(), and i3g4250d_filter_path_set().

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

- i3g4250d_reg.h

5.7 i3g4250d_fifo_ctrl_reg_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t fm: 3
- uint8_t wtm: 5

5.7.1 Field Documentation

5.7.1.1 fm

```
uint8_t i3g4250d_fifo_ctrl_reg_t::fm
```

Referenced by i3g4250d_fifo_mode_get(), and i3g4250d_fifo_mode_set().

5.7.1.2 wtm

```
uint8_t i3g4250d_fifo_ctrl_reg_t::wtm
```

Referenced by i3g4250d_fifo_watermark_get(), and i3g4250d_fifo_watermark_set().

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

- i3g4250d_reg.h

5.8 i3g4250d_fifo_src_reg_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t empty: 1
- uint8_t fss: 5
- uint8_t ovrn: 1
- uint8_t wtm: 1

5.8.1 Field Documentation

5.8.1.1 empty

```
uint8_t i3g4250d_fifo_src_reg_t::empty
```

Referenced by i3g4250d_fifo_empty_flag_get().

5.8.1.2 fss

```
uint8_t i3g4250d_fifo_src_reg_t::fss
```

Referenced by i3g4250d_fifo_data_level_get().

5.8.1.3 ovrn

```
uint8_t i3g4250d_fifo_src_reg_t::ovrn
```

Referenced by i3g4250d_fifo_ovr_flag_get().

5.8.1.4 wtm

```
uint8_t i3g4250d_fifo_src_reg_t::wtm
```

Referenced by i3g4250d_fifo_wtm_flag_get().

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

- i3g4250d_reg.h

5.9 i3g4250d_int1_cfg_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t and_or: 1
- uint8_t lir: 1
- uint8_t xhie: 1
- uint8_t xlie: 1
- uint8_t yhie: 1
- uint8_t ylie: 1
- uint8_t zhie: 1
- uint8_t zlie: 1

5.9.1 Field Documentation

5.9.1.1 and_or

```
uint8_t i3g4250d_int1_cfg_t::and_or
```

Referenced by i3g4250d_int_on_threshold_mode_get(), and i3g4250d_int_on_threshold_mode_set().

5.9.1.2 lir

```
uint8_t i3g4250d_int1_cfg_t::lir
```

Referenced by i3g4250d_int_notification_get(), and i3g4250d_int_notification_set().

5.9.1.3 xhie

```
uint8_t i3g4250d_int1_cfg_t::xhie
```

5.9.1.4 xlie

```
uint8_t i3g4250d_int1_cfg_t::xlie
```

5.9.1.5 yhie

```
uint8_t i3g4250d_int1_cfg_t::yhie
```

5.9.1.6 ylie

```
uint8_t i3g4250d_int1_cfg_t::ylie
```

5.9.1.7 zhie

```
uint8_t i3g4250d_int1_cfg_t::zhie
```

5.9.1.8 zlie

```
uint8_t i3g4250d_int1_cfg_t::zlie
```

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

- i3g4250d_reg.h

5.10 i3g4250d_int1_duration_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t d: 7
- uint8_t wait: 1

5.10.1 Field Documentation

5.10.1.1 d

```
uint8_t i3g4250d_int1_duration_t::d
```

Referenced by i3g4250d_int_on_threshold_dur_get(), and i3g4250d_int_on_threshold_dur_set().

5.10.1.2 wait

```
uint8_t i3g4250d_int1_duration_t::wait
```

Referenced by i3g4250d_int_on_threshold_dur_set().

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

- i3g4250d_reg.h

5.11 i3g4250d_int1_route_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t i1_boot: 1
- uint8_t i1_int1: 1

5.11.1 Field Documentation

5.11.1.1 i1_boot

```
uint8_t i3g4250d_int1_route_t::i1_boot
```

Referenced by i3g4250d_pin_int1_route_get(), and i3g4250d_pin_int1_route_set().

5.11.1.2 i1_int1

```
uint8_t i3g4250d_int1_route_t::i1_int1
```

Referenced by i3g4250d_pin_int1_route_get(), and i3g4250d_pin_int1_route_set().

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

- i3g4250d_reg.h

5.12 i3g4250d_int1_src_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t ia: 1
- uint8_t not_used_01: 1
- uint8_t xh: 1
- uint8_t xl: 1
- uint8_t yh: 1
- uint8_t yl: 1
- uint8_t zh: 1
- uint8_t zl: 1

5.12.1 Field Documentation

5.12.1.1 ia

```
uint8_t i3g4250d_int1_src_t::ia
```

5.12.1.2 not_used_01

```
uint8_t i3g4250d_int1_src_t::not_used_01
```

5.12.1.3 xh

```
uint8_t i3g4250d_int1_src_t::xh
```

5.12.1.4 xl

```
uint8_t i3g4250d_int1_src_t::xl
```

5.12.1.5 yh

```
uint8_t i3g4250d_int1_src_t::yh
```

5.12.1.6 yl

```
uint8_t i3g4250d_int1_src_t::yl
```

5.12.1.7 zh

```
uint8_t i3g4250d_int1_src_t::zh
```

5.12.1.8 zl

```
uint8_t i3g4250d_int1_src_t::zl
```

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

- i3g4250d_reg.h

5.13 i3g4250d_int1_tsh_xh_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t not_used_01: 1
- uint8_t thsx: 7

5.13.1 Field Documentation

5.13.1.1 not_used_01

```
uint8_t i3g4250d_int1_tsh_xh_t::not_used_01
```

5.13.1.2 thsx

```
uint8_t i3g4250d_int1_tsh_xh_t::thsx
```

Referenced by i3g4250d_int_x_threshold_get(), and i3g4250d_int_x_threshold_set().

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

- i3g4250d_reg.h

5.14 i3g4250d_int1_tsh_xl_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t thsx: 8

5.14.1 Field Documentation

5.14.1.1 thsx

```
uint8_t i3g4250d_int1_tsh_xl_t::thsx
```

Referenced by i3g4250d_int_x_threshold_set().

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

- i3g4250d_reg.h

5.15 i3g4250d_int1_tsh_yh_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t not_used_01: 1
- uint8_t thsy: 7

5.15.1 Field Documentation

5.15.1.1 not_used_01

```
uint8_t i3g4250d_int1_tsh_yh_t::not_used_01
```

5.15.1.2 thsy

```
uint8_t i3g4250d_int1_tsh_yh_t::thsy
```

Referenced by i3g4250d_int_y_threshold_get(), and i3g4250d_int_y_threshold_set().

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

- i3g4250d_reg.h

5.16 i3g4250d_int1_tsh_yl_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t thsy: 8

5.16.1 Field Documentation

5.16.1.1 thsy

```
uint8_t i3g4250d_int1_tsh_yl_t::thsy
```

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

- i3g4250d_reg.h

5.17 i3g4250d_int1_tsh_zh_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t not_used_01: 1
- uint8_t thsz: 7

5.17.1 Field Documentation

5.17.1.1 not_used_01

```
uint8_t i3g4250d_int1_tsh_zh_t::not_used_01
```

5.17.1.2 thsz

```
uint8_t i3g4250d_int1_tsh_zh_t::thsz
```

Referenced by i3g4250d_int_z_threshold_get(), and i3g4250d_int_z_threshold_set().

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

- i3g4250d_reg.h

5.18 i3g4250d_int1_tsh_zl_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t thsz: 8

5.18.1 Field Documentation

5.18.1.1 thsz

```
uint8_t i3g4250d_int1_tsh_zl_t::thsz
```

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

- i3g4250d_reg.h

5.19 i3g4250d_int2_route_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t i2_drdy: 1
- uint8_t i2_empty: 1
- uint8_t i2_orun: 1
- uint8_t i2_wtm: 1

5.19.1 Field Documentation

5.19.1.1 i2_drdy

```
uint8_t i3g4250d_int2_route_t::i2_drdy
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.19.1.2 i2_empty

```
uint8_t i3g4250d_int2_route_t::i2_empty
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.19.1.3 i2_orun

```
uint8_t i3g4250d_int2_route_t::i2_orun
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

5.19.1.4 i2_wtm

```
uint8_t i3g4250d_int2_route_t::i2_wtm
```

Referenced by i3g4250d_pin_int2_route_get(), and i3g4250d_pin_int2_route_set().

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

- i3g4250d_reg.h

5.20 i3g4250d_reference_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t ref: 8

5.20.1 Field Documentation

5.20.1.1 ref

```
uint8_t i3g4250d_reference_t::ref
```

Referenced by i3g4250d_hp_reference_value_get(), and i3g4250d_hp_reference_value_set().

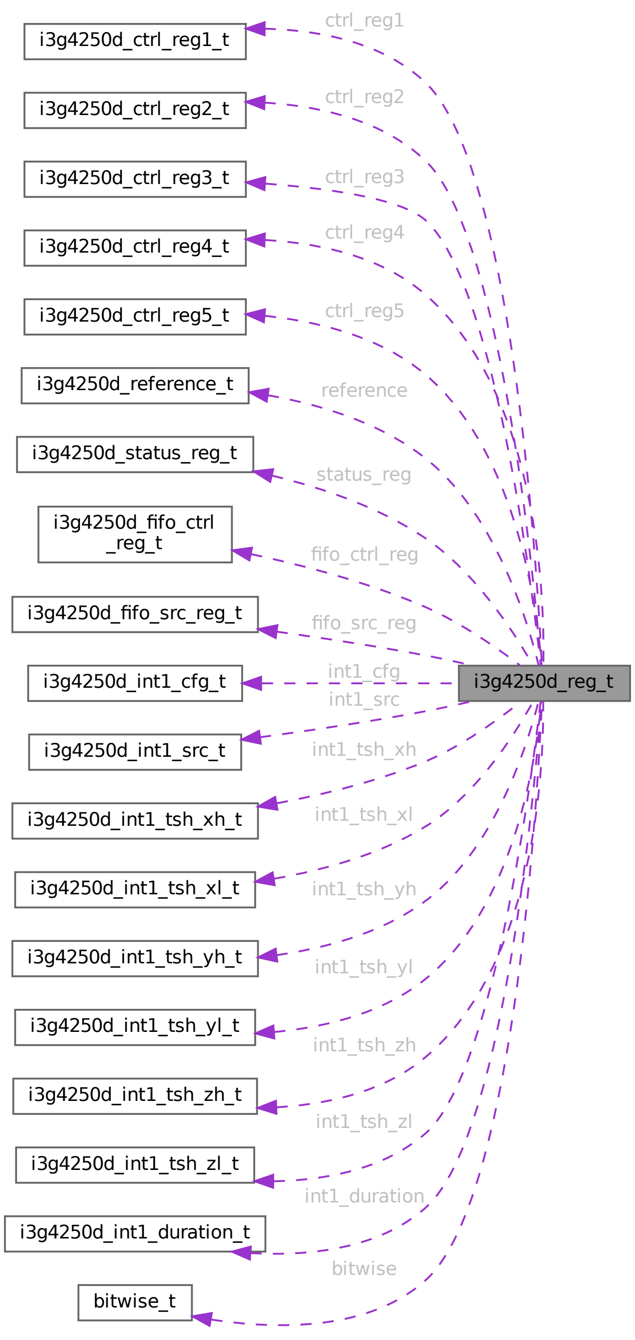
The documentation for this struct was generated from the following file:

- i3g4250d_reg.h

5.21 i3g4250d_reg_t Union Reference

```
#include <i3g4250d_reg.h>
```

Collaboration diagram for i3g4250d_reg_t:



Data Fields

- bitwise_t bitwise
- uint8_t byte
- i3g4250d_ctrl_reg1_t ctrl_reg1
- i3g4250d_ctrl_reg2_t ctrl_reg2
- i3g4250d_ctrl_reg3_t ctrl_reg3

- i3g4250d_ctrl_reg4_t ctrl_reg4
- i3g4250d_ctrl_reg5_t ctrl_reg5
- i3g4250d_fifo_ctrl_reg_t fifo_ctrl_reg
- i3g4250d_fifo_src_reg_t fifo_src_reg
- i3g4250d_int1_cfg_t int1_cfg
- i3g4250d_int1_duration_t int1_duration
- i3g4250d_int1_src_t int1_src
- i3g4250d_int1_tsh_xh_t int1_tsh_xh
- i3g4250d_int1_tsh_xl_t int1_tsh_xl
- i3g4250d_int1_tsh_yh_t int1_tsh_yh
- i3g4250d_int1_tsh_yl_t int1_tsh_yl
- i3g4250d_int1_tsh_zh_t int1_tsh_zh
- i3g4250d_int1_tsh_zl_t int1_tsh_zl
- i3g4250d_reference_t reference
- i3g4250d_status_reg_t status_reg

5.21.1 Field Documentation

5.21.1.1 bitwise

```
bitwise_t i3g4250d_reg_t::bitwise
```

5.21.1.2 byte

```
uint8_t i3g4250d_reg_t::byte
```

5.21.1.3 ctrl_reg1

```
i3g4250d_ctrl_reg1_t i3g4250d_reg_t::ctrl_reg1
```

5.21.1.4 ctrl_reg2

```
i3g4250d_ctrl_reg2_t i3g4250d_reg_t::ctrl_reg2
```

5.21.1.5 ctrl_reg3

```
i3g4250d_ctrl_reg3_t i3g4250d_reg_t::ctrl_reg3
```

5.21.1.6 ctrl_reg4

```
i3g4250d_ctrl_reg4_t i3g4250d_reg_t::ctrl_reg4
```

5.21.1.7 ctrl_reg5

```
i3g4250d_ctrl_reg5_t i3g4250d_reg_t::ctrl_reg5
```

5.21.1.8 fifo_ctrl_reg

`i3g4250d_fifo_ctrl_reg_t i3g4250d_reg_t::fifo_ctrl_reg`

5.21.1.9 fifo_src_reg

`i3g4250d_fifo_src_reg_t i3g4250d_reg_t::fifo_src_reg`

5.21.1.10 int1_cfg

`i3g4250d_int1_cfg_t i3g4250d_reg_t::int1_cfg`

5.21.1.11 int1_duration

`i3g4250d_int1_duration_t i3g4250d_reg_t::int1_duration`

5.21.1.12 int1_src

`i3g4250d_int1_src_t i3g4250d_reg_t::int1_src`

5.21.1.13 int1_tsh_xh

`i3g4250d_int1_tsh_xh_t i3g4250d_reg_t::int1_tsh_xh`

5.21.1.14 int1_tsh_xl

`i3g4250d_int1_tsh_xl_t i3g4250d_reg_t::int1_tsh_xl`

5.21.1.15 int1_tsh_yh

`i3g4250d_int1_tsh_yh_t i3g4250d_reg_t::int1_tsh_yh`

5.21.1.16 int1_tsh_yl

`i3g4250d_int1_tsh_yl_t i3g4250d_reg_t::int1_tsh_yl`

5.21.1.17 int1_tsh_zh

`i3g4250d_int1_tsh_zh_t i3g4250d_reg_t::int1_tsh_zh`

5.21.1.18 int1_tsh_zl

```
i3g4250d_int1_tsh_zl_t i3g4250d_reg_t::int1_tsh_zl
```

5.21.1.19 reference

```
i3g4250d_reference_t i3g4250d_reg_t::reference
```

5.21.1.20 status_reg

```
i3g4250d_status_reg_t i3g4250d_reg_t::status_reg
```

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

- i3g4250d_reg.h

5.22 i3g4250d_status_reg_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- uint8_t _xor: 1
- uint8_t xda: 1
- uint8_t yda: 1
- uint8_t yor: 1
- uint8_t zda: 1
- uint8_t zor: 1
- uint8_t zyxda: 1
- uint8_t zyxor: 1

5.22.1 Field Documentation

5.22.1.1 _xor

```
uint8_t i3g4250d_status_reg_t::_xor
```

5.22.1.2 xda

```
uint8_t i3g4250d_status_reg_t::xda
```

5.22.1.3 yda

```
uint8_t i3g4250d_status_reg_t::yda
```

5.22.1.4 yor

```
uint8_t i3g4250d_status_reg_t::yor
```

5.22.1.5 zda

```
uint8_t i3g4250d_status_reg_t::zda
```

5.22.1.6 zor

```
uint8_t i3g4250d_status_reg_t::zor
```

5.22.1.7 zyxda

```
uint8_t i3g4250d_status_reg_t::zyxda
```

Referenced by i3g4250d_flag_data_ready_get().

5.22.1.8 zyxor

```
uint8_t i3g4250d_status_reg_t::zyxor
```

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

- i3g4250d_reg.h

5.23 stmdev_ctx_t Struct Reference

```
#include <i3g4250d_reg.h>
```

Data Fields

- void ∗ handle
- stmdev_mdelay_ptr mdelay
- stmdev_read_ptr read_reg
- stmdev_write_ptr write_reg

5.23.1 Field Documentation

5.23.1.1 handle

```
void* stmdev_ctx_t::handle
```

Customizable optional pointer

Referenced by i3g4250d_read_reg(), i3g4250d_write_reg(), and main().

5.23.1.2 mdelay

`stmdev_mdelay_ptr stmdev_ctx_t::mdelay`

Component optional fields

Referenced by main().

5.23.1.3 read_reg

`stmdev_read_ptr stmdev_ctx_t::read_reg`

Referenced by i3g4250d_read_reg(), and main().

5.23.1.4 write_reg

`stmdev_write_ptr stmdev_ctx_t::write_reg`

Component mandatory fields

Referenced by i3g4250d_write_reg(), and main().

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

- i3g4250d_reg.h

5.24 ucf_line_t Struct Reference

`#include <i3g4250d_reg.h>`

Data Fields

- uint8_t address
- uint8_t data

5.24.1 Field Documentation

5.24.1.1 address

`uint8_t ucf_line_t::address`

5.24.1.2 data

`uint8_t ucf_line_t::data`

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

- i3g4250d_reg.h

Chapter 6

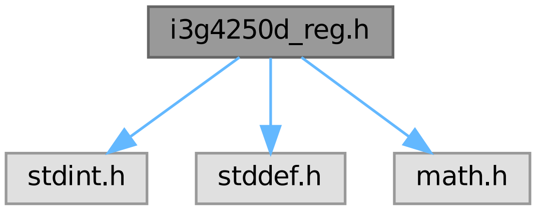
File Documentation

6.1 i3g4250d_reg.h File Reference

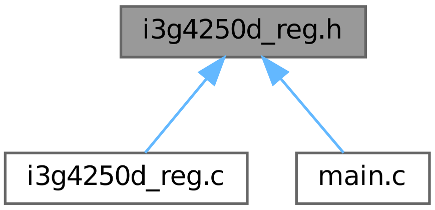
This file contains all the functions prototypes for the i3g4250d_reg.c driver.

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

Include dependency graph for i3g4250d_reg.h:



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



Data Structures

- struct bitwise_t
- struct i3g4250d_ctrl_reg1_t
- struct i3g4250d_ctrl_reg2_t
- struct i3g4250d_ctrl_reg3_t
- struct i3g4250d_ctrl_reg4_t
- struct i3g4250d_ctrl_reg5_t
- struct i3g4250d_fifo_ctrl_reg_t
- struct i3g4250d_fifo_src_reg_t
- struct i3g4250d_int1_cfg_t
- struct i3g4250d_int1_duration_t
- struct i3g4250d_int1_route_t
- struct i3g4250d_int1_src_t
- struct i3g4250d_int1_tsh_xh_t
- struct i3g4250d_int1_tsh_xl_t
- struct i3g4250d_int1_tsh_yh_t
- struct i3g4250d_int1_tsh_yl_t
- struct i3g4250d_int1_tsh_zh_t
- struct i3g4250d_int1_tsh_zl_t
- struct i3g4250d_int2_route_t
- struct i3g4250d_reference_t
- union i3g4250d_reg_t
- struct i3g4250d_status_reg_t
- struct stmdev_ctx_t
- struct ucf_line_t

Macros

- #define __weak __attribute__((weak))
- #define DRV_BIG_ENDIAN 4321
- #define DRV_BYTE_ORDER DRV_LITTLE_ENDIAN
- #define DRV_LITTLE_ENDIAN 1234
- #define I3G4250D_CTRL_REG1 0x20U
- #define I3G4250D_CTRL_REG2 0x21U
- #define I3G4250D_CTRL_REG3 0x22U
- #define I3G4250D_CTRL_REG4 0x23U
- #define I3G4250D_CTRL_REG5 0x24U
- #define I3G4250D_FIFO_CTRL_REG 0x2EU
- #define I3G4250D_FIFO_SRC_REG 0x2FU
- #define I3G4250D_I2C_ADD_H 0xD3U
- #define I3G4250D_I2C_ADD_L 0xD1U
- #define I3G4250D_ID 0xD3U
- #define I3G4250D_INT1_CFG 0x30U
- #define I3G4250D_INT1_DURATION 0x38U
- #define I3G4250D_INT1_SRC 0x31U
- #define I3G4250D_INT1_TSH_XH 0x32U
- #define I3G4250D_INT1_TSH_XL 0x33U
- #define I3G4250D_INT1_TSH_YH 0x34U
- #define I3G4250D_INT1_TSH_YL 0x35U
- #define I3G4250D_INT1_TSH_ZH 0x36U
- #define I3G4250D_INT1_TSH_ZL 0x37U
- #define I3G4250D_OUT_TEMP 0x26U
- #define I3G4250D_OUT_X_H 0x29U

- #define I3G4250D_OUT_X_L 0x28U
- #define I3G4250D_OUT_Y_H 0x2BU
- #define I3G4250D_OUT_Y_L 0x2AU
- #define I3G4250D_OUT_Z_H 0x2DU
- #define I3G4250D_OUT_Z_L 0x2CU
- #define I3G4250D_REFERENCE 0x25U
- #define I3G4250D_STATUS_REG 0x27U
- #define I3G4250D_WHO_AM_I 0x0FU
- #define MEMS_SHARED_TYPES
- #define MEMS_UCF_SHARED_TYPES
- #define PROPERTY_DISABLE (0U)
- #define PROPERTY_ENABLE (1U)

Typedefs

- typedef void(∗ stmdev_mdelay_ptr) (uint32_t millisec)
- typedef int32_t(∗ stmdev_read_ptr) (void ∗, uint8_t, uint8_t ∗, uint16_t)
- typedef int32_t(∗ stmdev_write_ptr) (void ∗, uint8_t, const uint8_t ∗, uint16_t)

Enumerations

- enum i3g4250d_and_or_t { I3G4250D_INT1_ON_TH_AND = 1 , I3G4250D_INT1_ON_TH_OR = 0 }
- enum i3g4250d_ble_t { I3G4250D_AUX_LSB_AT_LOW_ADD = 0 , I3G4250D_AUX_MSB_AT_LOW_ADD = 1 }
- enum i3g4250d_bw_t { I3G4250D_CUT_OFF_LOW = 0 , I3G4250D_CUT_OFF_MEDIUM = 1 , I3G4250D_CUT_OFF_HIGH = 2 , I3G4250D_CUT_OFF_VERY_HIGH = 3 }
- enum i3g4250d_dr_t {
  I3G4250D_ODR_OFF = 0x00 , I3G4250D_ODR_SLEEP = 0x08 , I3G4250D_ODR_100Hz = 0x0F , I3G4250D_ODR_200Hz = 0x1F ,
  I3G4250D_ODR_400Hz = 0x2F , I3G4250D_ODR_800Hz = 0x3F }
- enum i3g4250d_fifo_mode_t { I3G4250D_FIFO_BYPASS_MODE = 0x00 , I3G4250D_FIFO_MODE = 0x01 , I3G4250D_FIFO_STREAM_MODE = 0x02 }
- enum i3g4250d_fs_t { I3G4250D_245dps = 0x00 , I3G4250D_500dps = 0x01 , I3G4250D_2000dps = 0x02 }
- enum i3g4250d_h_lactive_t { I3G4250D_ACTIVE_HIGH = 0 , I3G4250D_ACTIVE_LOW = 1 }
- enum i3g4250d_hpcf_t {
  I3G4250D_HP_LEVEL_0 = 0 , I3G4250D_HP_LEVEL_1 = 1 , I3G4250D_HP_LEVEL_2 = 2 , I3G4250D_HP_LEVEL_3 = 3 ,
  I3G4250D_HP_LEVEL_4 = 4 , I3G4250D_HP_LEVEL_5 = 5 , I3G4250D_HP_LEVEL_6 = 6 , I3G4250D_HP_LEVEL_7 = 7 ,
  I3G4250D_HP_LEVEL_8 = 8 , I3G4250D_HP_LEVEL_9 = 9 }
- enum i3g4250d_hpm_t { I3G4250D_HP_NORMAL_MODE_WITH_RST = 0 , I3G4250D_HP_REFERENCE_SIGNAL = 1 , I3G4250D_HP_NORMAL_MODE = 2 , I3G4250D_HP_AUTO_RESET_ON_INT = 3 }
- enum i3g4250d_int1_sel_t { I3G4250D_ONLY_LPF1_ON_INT = 0 , I3G4250D_LPF1_HP_ON_INT = 1 , I3G4250D_LPF1_LPF2_ON_INT = 2 , I3G4250D_LPF1_HP_LPF2_ON_INT = 6 }
- enum i3g4250d_lir_t { I3G4250D_INT_PULSED = 0 , I3G4250D_INT_LATCHED = 1 }
- enum i3g4250d_out_sel_t { I3G4250D_ONLY_LPF1_ON_OUT = 0 , I3G4250D_LPF1_HP_ON_OUT = 1 , I3G4250D_LPF1_LPF2_ON_OUT = 2 , I3G4250D_LPF1_HP_LPF2_ON_OUT = 6 }
- enum i3g4250d_pp_od_t { I3G4250D_PUSH_PULL = 0 , I3G4250D_OPEN_DRAIN = 1 }
- enum i3g4250d_sim_t { I3G4250D_SPI_4_WIRE = 0 , I3G4250D_SPI_3_WIRE = 1 }
- enum i3g4250d_st_t { I3G4250D_GY_ST_DISABLE = 0 , I3G4250D_GY_ST_POSITIVE = 1 , I3G4250D_GY_ST_NEGATIVE = 3 }

Functions

- int32_t i3g4250d_angular_rate_raw_get (const stmdev_ctx_t ∗ctx, int16_t ∗val)

  Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].
- int32_t i3g4250d_axis_x_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_x_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_axis_y_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_y_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_axis_z_data_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)
- int32_t i3g4250d_axis_z_data_set (const stmdev_ctx_t ∗ctx, uint8_t val)
- int32_t i3g4250d_boot_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reboot memory content. Reload the calibration parameters.[get].
- int32_t i3g4250d_boot_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reboot memory content. Reload the calibration parameters.[set].
- int32_t i3g4250d_data_format_get (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t ∗val)

  Big/Little Endian data selection.[get].
- int32_t i3g4250d_data_format_set (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t val)

  Big/Little Endian data selection.[set].
- int32_t i3g4250d_data_rate_get (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t ∗val)

  Accelerometer data rate selection.[get].
- int32_t i3g4250d_data_rate_set (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t val)

  Accelerometer data rate selection.[set].
- int32_t i3g4250d_device_id_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Device Who amI.[get].
- int32_t i3g4250d_fifo_data_level_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO stored data level[get].
- int32_t i3g4250d_fifo_empty_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOemptybit.[get].
- int32_t i3g4250d_fifo_enable_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOenable.[get].
- int32_t i3g4250d_fifo_enable_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFOenable.[set].
- int32_t i3g4250d_fifo_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t ∗val)

  FIFO mode selection.[get].
- int32_t i3g4250d_fifo_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t val)

  FIFO mode selection.[set].
- int32_t i3g4250d_fifo_ovr_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Overrun bit status.[get].
- int32_t i3g4250d_fifo_watermark_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO watermark level selection.[get].
- int32_t i3g4250d_fifo_watermark_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFO watermark level selection.[set].
- int32_t i3g4250d_fifo_wtm_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)
- int32_t i3g4250d_filter_path_get (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t ∗val)

  Out/FIFO selection path. [get].
- int32_t i3g4250d_filter_path_internal_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t ∗val)

  Interrupt generator selection path.[get].
- int32_t i3g4250d_filter_path_internal_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t val)

  Interrupt generator selection path.[set].
- int32_t i3g4250d_filter_path_set (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t val)

Out/FIFO selection path. [set].
- int32_t i3g4250d_flag_data_ready_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Accelerometer new data available.[get].
- float_t i3g4250d_from_fs245dps_to_mdps (int16_t lsb)
- float_t i3g4250d_from_lsb_to_celsius (int16_t lsb)
- int32_t i3g4250d_full_scale_get (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t ∗val)

  Gyroscope full-scale selection.[get].
- int32_t i3g4250d_full_scale_set (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t val)

  Gyroscope full-scale selection.[set].
- int32_t i3g4250d_hp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t ∗val)

  High-pass filter bandwidth selection.[get].
- int32_t i3g4250d_hp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t val)

  High-pass filter bandwidth selection.[set].
- int32_t i3g4250d_hp_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t ∗val)

  High-pass filter mode selection. [get].
- int32_t i3g4250d_hp_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t val)

  High-pass filter mode selection. [set].
- int32_t i3g4250d_hp_reference_value_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reference value for high-pass filter.[get].
- int32_t i3g4250d_hp_reference_value_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reference value for high-pass filter.[set].
- int32_t i3g4250d_int_notification_get (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t ∗val)

  Latched/pulsed interrupt.[get].
- int32_t i3g4250d_int_notification_set (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t val)

  Latched/pulsed interrupt.[set].
- int32_t i3g4250d_int_on_threshold_conf_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[get].
- int32_t i3g4250d_int_on_threshold_conf_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

  Configure the interrupt threshold sign.[set].
- int32_t i3g4250d_int_on_threshold_dur_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Durationvalue.[get].
- int32_t i3g4250d_int_on_threshold_dur_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Durationvalue.[set].
- int32_t i3g4250d_int_on_threshold_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t ∗val)

  AND/OR combination of interrupt events.[get].
- int32_t i3g4250d_int_on_threshold_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t val)

  AND/OR combination of interrupt events.[set].
- int32_t i3g4250d_int_on_threshold_src_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_src_t ∗val)

  int_on_threshold_src: [get]
- int32_t i3g4250d_int_x_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on X.[get].
- int32_t i3g4250d_int_x_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on X.[set].
- int32_t i3g4250d_int_y_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Y.[get].
- int32_t i3g4250d_int_y_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Y.[set].
- int32_t i3g4250d_int_z_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

  Interrupt threshold on Z.[get].
- int32_t i3g4250d_int_z_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

  Interrupt threshold on Z.[set].

- int32_t i3g4250d_lp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t ∗val)

    Lowpass filter bandwidth selection.[get].
- int32_t i3g4250d_lp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t val)

    Lowpass filter bandwidth selection.[set].
- int32_t i3g4250d_pin_int1_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t ∗val)

    Select the signal that need to route on int1 pad.[get].
- int32_t i3g4250d_pin_int1_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t val)

    Select the signal that need to route on int1 pad.[set].
- int32_t i3g4250d_pin_int2_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t ∗val)

    Select the signal that need to route on int2 pad.[get].
- int32_t i3g4250d_pin_int2_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t val)

    Select the signal that need to route on int2 pad.[set].
- int32_t i3g4250d_pin_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t ∗val)

    Push-pull/open drain selection on interrupt pads.[get].
- int32_t i3g4250d_pin_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t val)

    Push-pull/open drain selection on interrupt pads.[set].
- int32_t i3g4250d_pin_polarity_get (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t ∗val)

    Pin active-high/low.[get].
- int32_t i3g4250d_pin_polarity_set (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t val)

    Pin active-high/low.[set].
- int32_t i3g4250d_read_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

    Read generic device register.
- int32_t i3g4250d_self_test_get (const stmdev_ctx_t ∗ctx, i3g4250d_st_t ∗val)

    Angular rate sensor self-test enable. [get].
- int32_t i3g4250d_self_test_set (const stmdev_ctx_t ∗ctx, i3g4250d_st_t val)

    Angular rate sensor self-test enable. [set].
- int32_t i3g4250d_spi_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t ∗val)

    SPI Serial Interface Mode selection.[get].
- int32_t i3g4250d_spi_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t val)

    SPI Serial Interface Mode selection.[set].
- int32_t i3g4250d_status_reg_get (const stmdev_ctx_t ∗ctx, i3g4250d_status_reg_t ∗val)

    The STATUS_REG register is read by the primary interface.[get].
- int32_t i3g4250d_temperature_raw_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

    Temperature data.[get].
- int32_t i3g4250d_write_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

    Write generic device register.

6.1.1 Detailed Description

This file contains all the functions prototypes for the i3g4250d_reg.c driver.

Author

Sensors Software Solution Team

Attention

6.2 i3g4250d_reg.h

Go to the documentation of this file.

```
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef I3G4250D_REGS_H
#define I3G4250D_REGS_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include <stdint.h>
#include <stddef.h>
#include <math.h>

#ifndef __BYTE_ORDER__

#define DRV_LITTLE_ENDIAN 1234
#define DRV_BIG_ENDIAN    4321

//#define DRV_BYTE_ORDER         DRV_BIG_ENDIAN
#define DRV_BYTE_ORDER         DRV_LITTLE_ENDIAN

#else /* defined __BYTE_ORDER__ */

#define DRV_LITTLE_ENDIAN __ORDER_LITTLE_ENDIAN__
#define DRV_BIG_ENDIAN    __ORDER_BIG_ENDIAN__
#define DRV_BYTE_ORDER    __BYTE_ORDER__

#endif /* DRV_BYTE_ORDER */

#ifndef MEMS_SHARED_TYPES
#define MEMS_SHARED_TYPES

typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t bit0       : 1;
  uint8_t bit1       : 1;
  uint8_t bit2       : 1;
  uint8_t bit3       : 1;
  uint8_t bit4       : 1;
  uint8_t bit5       : 1;
  uint8_t bit6       : 1;
  uint8_t bit7       : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t bit7       : 1;
  uint8_t bit6       : 1;
  uint8_t bit5       : 1;
  uint8_t bit4       : 1;
  uint8_t bit3       : 1;
  uint8_t bit2       : 1;
  uint8_t bit1       : 1;
  uint8_t bit0       : 1;
#endif /* DRV_BYTE_ORDER */
} bitwise_t;

#define PROPERTY_DISABLE                (0U)
#define PROPERTY_ENABLE                 (1U)

typedef int32_t (*stmdev_write_ptr)(void *, uint8_t, const uint8_t *, uint16_t);
typedef int32_t (*stmdev_read_ptr)(void *, uint8_t, uint8_t *, uint16_t);
typedef void (*stmdev_mdelay_ptr)(uint32_t millisec);

typedef struct
{
  stmdev_write_ptr  write_reg;
  stmdev_read_ptr   read_reg;
  stmdev_mdelay_ptr   mdelay;
  void *handle;
} stmdev_ctx_t;

#endif /* MEMS_SHARED_TYPES */

#ifndef MEMS_UCF_SHARED_TYPES
#define MEMS_UCF_SHARED_TYPES

typedef struct
{
  uint8_t address;
  uint8_t data;
} ucf_line_t;

#endif /* MEMS_UCF_SHARED_TYPES */

#define I3G4250D_I2C_ADD_L               0xD1U
#define I3G4250D_I2C_ADD_H               0xD3U

#define I3G4250D_ID                      0xD3U

#define I3G4250D_WHO_AM_I                0x0FU
#define I3G4250D_CTRL_REG1               0x20U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t pd                         : 4; /* xen yen zen pd */
  uint8_t bw                         : 2;
  uint8_t dr                         : 2;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t dr                         : 2;
  uint8_t bw                         : 2;
  uint8_t pd                         : 4; /* xen yen zen pd */
#endif /* DRV_BYTE_ORDER */
} i3g4250d_ctrl_reg1_t;

#define I3G4250D_CTRL_REG2               0x21U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t hpcf                       : 4;
  uint8_t hpm                        : 2;
  uint8_t not_used_01                : 2;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_01                : 2;
  uint8_t hpm                        : 2;
  uint8_t hpcf                       : 4;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_ctrl_reg2_t;

#define I3G4250D_CTRL_REG3               0x22U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t i2_empty                   : 1;
  uint8_t i2_orun                    : 1;
  uint8_t i2_wtm                     : 1;
  uint8_t i2_drdy                    : 1;
  uint8_t pp_od                      : 1;
  uint8_t h_lactive                  : 1;
  uint8_t i1_boot                    : 1;
  uint8_t i1_int1                    : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t i1_int1                    : 1;
  uint8_t i1_boot                    : 1;
  uint8_t h_lactive                  : 1;
  uint8_t pp_od                      : 1;
  uint8_t i2_drdy                    : 1;
  uint8_t i2_wtm                     : 1;
  uint8_t i2_orun                    : 1;
  uint8_t i2_empty                   : 1;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_ctrl_reg3_t;

#define I3G4250D_CTRL_REG4               0x23U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t sim                        : 1;
  uint8_t st                         : 2;
  uint8_t not_used_01                : 1;
  uint8_t fs                         : 2;
  uint8_t ble                        : 1;
  uint8_t not_used_02                : 1;
```

```
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_02                 : 1;
  uint8_t ble                         : 1;
  uint8_t fs                          : 2;
  uint8_t not_used_01                 : 1;
  uint8_t st                          : 2;
  uint8_t sim                         : 1;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_ctrl_reg4_t;

#define I3G4250D_CTRL_REG5               0x24U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t out_sel                     : 2;
  uint8_t int1_sel                    : 2;
  uint8_t hpen                        : 1;
  uint8_t not_used_01                 : 1;
  uint8_t fifo_en                     : 1;
  uint8_t boot                        : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t boot                        : 1;
  uint8_t fifo_en                     : 1;
  uint8_t not_used_01                 : 1;
  uint8_t hpen                        : 1;
  uint8_t int1_sel                    : 2;
  uint8_t out_sel                     : 2;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_ctrl_reg5_t;

#define I3G4250D_REFERENCE               0x25U
typedef struct
{
  uint8_t ref                         : 8;
} i3g4250d_reference_t;

#define I3G4250D_OUT_TEMP                0x26U
#define I3G4250D_STATUS_REG              0x27U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t xda                         : 1;
  uint8_t yda                         : 1;
  uint8_t zda                         : 1;
  uint8_t zyxda                       : 1;
  uint8_t _xor                        : 1;
  uint8_t yor                         : 1;
  uint8_t zor                         : 1;
  uint8_t zyxor                       : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t zyxor                       : 1;
  uint8_t zor                         : 1;
  uint8_t yor                         : 1;
  uint8_t _xor                        : 1;
  uint8_t zyxda                       : 1;
  uint8_t zda                         : 1;
  uint8_t yda                         : 1;
  uint8_t xda                         : 1;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_status_reg_t;

#define I3G4250D_OUT_X_L                 0x28U
#define I3G4250D_OUT_X_H                 0x29U
#define I3G4250D_OUT_Y_L                 0x2AU
#define I3G4250D_OUT_Y_H                 0x2BU
#define I3G4250D_OUT_Z_L                 0x2CU
#define I3G4250D_OUT_Z_H                 0x2DU
#define I3G4250D_FIFO_CTRL_REG           0x2EU
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t wtm                         : 5;
  uint8_t fm                          : 3;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t fm                          : 3;
  uint8_t wtm                         : 5;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_fifo_ctrl_reg_t;

#define I3G4250D_FIFO_SRC_REG            0x2FU
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t fss                         : 5;
  uint8_t empty                       : 1;
  uint8_t ovrn                        : 1;
  uint8_t wtm                         : 1;
```

```
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t wtm                            : 1;
  uint8_t ovrn                           : 1;
  uint8_t empty                          : 1;
  uint8_t fss                            : 5;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_fifo_src_reg_t;

#define I3G4250D_INT1_CFG                0x30U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t xlie                           : 1;
  uint8_t xhie                           : 1;
  uint8_t ylie                           : 1;
  uint8_t yhie                           : 1;
  uint8_t zlie                           : 1;
  uint8_t zhie                           : 1;
  uint8_t lir                            : 1;
  uint8_t and_or                         : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t and_or                         : 1;
  uint8_t lir                            : 1;
  uint8_t zhie                           : 1;
  uint8_t zlie                           : 1;
  uint8_t yhie                           : 1;
  uint8_t ylie                           : 1;
  uint8_t xhie                           : 1;
  uint8_t xlie                           : 1;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_cfg_t;

#define I3G4250D_INT1_SRC                0x31U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t xl                             : 1;
  uint8_t xh                             : 1;
  uint8_t yl                             : 1;
  uint8_t yh                             : 1;
  uint8_t zl                             : 1;
  uint8_t zh                             : 1;
  uint8_t ia                             : 1;
  uint8_t not_used_01                    : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_01                    : 1;
  uint8_t ia                             : 1;
  uint8_t zh                             : 1;
  uint8_t zl                             : 1;
  uint8_t yh                             : 1;
  uint8_t yl                             : 1;
  uint8_t xh                             : 1;
  uint8_t xl                             : 1;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_src_t;

#define I3G4250D_INT1_TSH_XH             0x32U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t thsx                           : 7;
  uint8_t not_used_01                    : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_01                    : 1;
  uint8_t thsx                           : 7;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_tsh_xh_t;

#define I3G4250D_INT1_TSH_XL             0x33U
typedef struct
{
  uint8_t thsx                           : 8;
} i3g4250d_int1_tsh_xl_t;

#define I3G4250D_INT1_TSH_YH             0x34U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t thsy                           : 7;
  uint8_t not_used_01                    : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_01                    : 1;
  uint8_t thsy                           : 7;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_tsh_yh_t;

#define I3G4250D_INT1_TSH_YL             0x35U
```

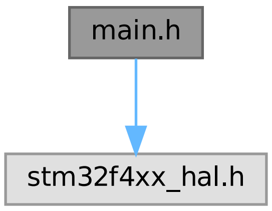
```
typedef struct
{
  uint8_t thsy                          : 8;
} i3g4250d_int1_tsh_yl_t;

#define I3G4250D_INT1_TSH_ZH             0x36U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t thsz                          : 7;
  uint8_t not_used_01                   : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t not_used_01                   : 1;
  uint8_t thsz                          : 7;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_tsh_zh_t;

#define I3G4250D_INT1_TSH_ZL             0x37U
typedef struct
{
  uint8_t thsz                          : 8;
} i3g4250d_int1_tsh_zl_t;

#define I3G4250D_INT1_DURATION           0x38U
typedef struct
{
#if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
  uint8_t d                             : 7;
  uint8_t wait                          : 1;
#elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
  uint8_t wait                          : 1;
  uint8_t d                             : 7;
#endif /* DRV_BYTE_ORDER */
} i3g4250d_int1_duration_t;

typedef union
{
  i3g4250d_ctrl_reg1_t        ctrl_reg1;
  i3g4250d_ctrl_reg2_t        ctrl_reg2;
  i3g4250d_ctrl_reg3_t        ctrl_reg3;
  i3g4250d_ctrl_reg4_t        ctrl_reg4;
  i3g4250d_ctrl_reg5_t        ctrl_reg5;
  i3g4250d_reference_t        reference;
  i3g4250d_status_reg_t       status_reg;
  i3g4250d_fifo_ctrl_reg_t    fifo_ctrl_reg;
  i3g4250d_fifo_src_reg_t     fifo_src_reg;
  i3g4250d_int1_cfg_t         int1_cfg;
  i3g4250d_int1_src_t         int1_src;
  i3g4250d_int1_tsh_xh_t      int1_tsh_xh;
  i3g4250d_int1_tsh_xl_t      int1_tsh_xl;
  i3g4250d_int1_tsh_yh_t      int1_tsh_yh;
  i3g4250d_int1_tsh_yl_t      int1_tsh_yl;
  i3g4250d_int1_tsh_zh_t      int1_tsh_zh;
  i3g4250d_int1_tsh_zl_t      int1_tsh_zl;
  i3g4250d_int1_duration_t    int1_duration;
  bitwise_t                   bitwise;
  uint8_t                     byte;
} i3g4250d_reg_t;

#ifndef __weak
#define __weak __attribute__((weak))
#endif /* __weak */

/*
 * These are the basic platform dependent I/O routines to read
 * and write device registers connected on a standard bus.
 * The driver keeps offering a default implementation based on function
 * pointers to read/write routines for backward compatibility.
 * The __weak directive allows the final application to overwrite
 * them with a custom implementation.
 */

int32_t i3g4250d_read_reg(const stmdev_ctx_t *ctx, uint8_t reg,
                          uint8_t *data,
                          uint16_t len);
int32_t i3g4250d_write_reg(const stmdev_ctx_t *ctx, uint8_t reg,
                           uint8_t *data,
                           uint16_t len);

float_t i3g4250d_from_fs245dps_to_mdps(int16_t lsb);
float_t i3g4250d_from_lsb_to_celsius(int16_t lsb);

int32_t i3g4250d_axis_x_data_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_axis_x_data_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_axis_y_data_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_axis_y_data_get(const stmdev_ctx_t *ctx, uint8_t *val);
```

```

int32_t i3g4250d_axis_z_data_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_axis_z_data_get(const stmdev_ctx_t *ctx, uint8_t *val);

typedef enum
{
  I3G4250D_ODR_OFF     = 0x00,
  I3G4250D_ODR_SLEEP   = 0x08,
  I3G4250D_ODR_100Hz   = 0x0F,
  I3G4250D_ODR_200Hz   = 0x1F,
  I3G4250D_ODR_400Hz   = 0x2F,
  I3G4250D_ODR_800Hz   = 0x3F,
} i3g4250d_dr_t;
int32_t i3g4250d_data_rate_set(const stmdev_ctx_t *ctx, i3g4250d_dr_t val);
int32_t i3g4250d_data_rate_get(const stmdev_ctx_t *ctx, i3g4250d_dr_t *val);

typedef enum
{
  I3G4250D_245dps     = 0x00,
  I3G4250D_500dps     = 0x01,
  I3G4250D_2000dps    = 0x02,
} i3g4250d_fs_t;
int32_t i3g4250d_full_scale_set(const stmdev_ctx_t *ctx, i3g4250d_fs_t val);
int32_t i3g4250d_full_scale_get(const stmdev_ctx_t *ctx,
                                i3g4250d_fs_t *val);

int32_t i3g4250d_status_reg_get(const stmdev_ctx_t *ctx,
                                i3g4250d_status_reg_t *val);

int32_t i3g4250d_flag_data_ready_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_temperature_raw_get(const stmdev_ctx_t *ctx,
                                     uint8_t *buff);

int32_t i3g4250d_angular_rate_raw_get(const stmdev_ctx_t *ctx,
                                      int16_t *val);

int32_t i3g4250d_device_id_get(const stmdev_ctx_t *ctx, uint8_t *buff);

typedef enum
{
  I3G4250D_GY_ST_DISABLE    = 0,
  I3G4250D_GY_ST_POSITIVE   = 1,
  I3G4250D_GY_ST_NEGATIVE   = 3,
} i3g4250d_st_t;
int32_t i3g4250d_self_test_set(const stmdev_ctx_t *ctx, i3g4250d_st_t val);
int32_t i3g4250d_self_test_get(const stmdev_ctx_t *ctx, i3g4250d_st_t *val);

typedef enum
{
  I3G4250D_AUX_LSB_AT_LOW_ADD   = 0,
  I3G4250D_AUX_MSB_AT_LOW_ADD   = 1,
} i3g4250d_ble_t;
int32_t i3g4250d_data_format_set(const stmdev_ctx_t *ctx,
                                 i3g4250d_ble_t val);
int32_t i3g4250d_data_format_get(const stmdev_ctx_t *ctx,
                                 i3g4250d_ble_t *val);

int32_t i3g4250d_boot_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_boot_get(const stmdev_ctx_t *ctx, uint8_t *val);

typedef enum
{
  I3G4250D_CUT_OFF_LOW        = 0,
  I3G4250D_CUT_OFF_MEDIUM     = 1,
  I3G4250D_CUT_OFF_HIGH       = 2,
  I3G4250D_CUT_OFF_VERY_HIGH  = 3,
} i3g4250d_bw_t;
int32_t i3g4250d_lp_bandwidth_set(const stmdev_ctx_t *ctx,
                                  i3g4250d_bw_t val);
int32_t i3g4250d_lp_bandwidth_get(const stmdev_ctx_t *ctx,
                                  i3g4250d_bw_t *val);

typedef enum
{
  I3G4250D_HP_LEVEL_0   = 0,
  I3G4250D_HP_LEVEL_1   = 1,
  I3G4250D_HP_LEVEL_2   = 2,
  I3G4250D_HP_LEVEL_3   = 3,
  I3G4250D_HP_LEVEL_4   = 4,
  I3G4250D_HP_LEVEL_5   = 5,
  I3G4250D_HP_LEVEL_6   = 6,
  I3G4250D_HP_LEVEL_7   = 7,
  I3G4250D_HP_LEVEL_8   = 8,
  I3G4250D_HP_LEVEL_9   = 9,
} i3g4250d_hpcf_t;
int32_t i3g4250d_hp_bandwidth_set(const stmdev_ctx_t *ctx,
                                   i3g4250d_hpcf_t val);
int32_t i3g4250d_hp_bandwidth_get(const stmdev_ctx_t *ctx,
                                   i3g4250d_hpcf_t *val);

typedef enum
{
  I3G4250D_HP_NORMAL_MODE_WITH_RST  = 0,
  I3G4250D_HP_REFERENCE_SIGNAL      = 1,
  I3G4250D_HP_NORMAL_MODE           = 2,
  I3G4250D_HP_AUTO_RESET_ON_INT     = 3,
} i3g4250d_hpm_t;
int32_t i3g4250d_hp_mode_set(const stmdev_ctx_t *ctx, i3g4250d_hpm_t val);
int32_t i3g4250d_hp_mode_get(const stmdev_ctx_t *ctx, i3g4250d_hpm_t *val);

typedef enum
{
  I3G4250D_ONLY_LPF1_ON_OUT     = 0,
  I3G4250D_LPF1_HP_ON_OUT       = 1,
  I3G4250D_LPF1_LPF2_ON_OUT     = 2,
  I3G4250D_LPF1_HP_LPF2_ON_OUT  = 6,
} i3g4250d_out_sel_t;
int32_t i3g4250d_filter_path_set(const stmdev_ctx_t *ctx,
                                  i3g4250d_out_sel_t val);
int32_t i3g4250d_filter_path_get(const stmdev_ctx_t *ctx,
                                  i3g4250d_out_sel_t *val);

typedef enum
{
  I3G4250D_ONLY_LPF1_ON_INT     = 0,
  I3G4250D_LPF1_HP_ON_INT       = 1,
  I3G4250D_LPF1_LPF2_ON_INT     = 2,
  I3G4250D_LPF1_HP_LPF2_ON_INT  = 6,
} i3g4250d_int1_sel_t;
int32_t i3g4250d_filter_path_internal_set(const stmdev_ctx_t *ctx,
                                           i3g4250d_int1_sel_t val);
int32_t i3g4250d_filter_path_internal_get(const stmdev_ctx_t *ctx,
                                           i3g4250d_int1_sel_t *val);

int32_t i3g4250d_hp_reference_value_set(const stmdev_ctx_t *ctx,
                                         uint8_t val);
int32_t i3g4250d_hp_reference_value_get(const stmdev_ctx_t *ctx,
                                         uint8_t *val);

typedef enum
{
  I3G4250D_SPI_4_WIRE   = 0,
  I3G4250D_SPI_3_WIRE   = 1,
} i3g4250d_sim_t;
int32_t i3g4250d_spi_mode_set(const stmdev_ctx_t *ctx, i3g4250d_sim_t val);
int32_t i3g4250d_spi_mode_get(const stmdev_ctx_t *ctx, i3g4250d_sim_t *val);

typedef struct
{
  uint8_t i1_int1              : 1;
  uint8_t i1_boot              : 1;
} i3g4250d_int1_route_t;
int32_t i3g4250d_pin_int1_route_set(const stmdev_ctx_t *ctx,
                                     i3g4250d_int1_route_t val);
int32_t i3g4250d_pin_int1_route_get(const stmdev_ctx_t *ctx,
                                     i3g4250d_int1_route_t *val);

typedef struct
{
  uint8_t i2_empty              : 1;
  uint8_t i2_orun               : 1;
  uint8_t i2_wtm                : 1;
  uint8_t i2_drdy               : 1;
} i3g4250d_int2_route_t;
int32_t i3g4250d_pin_int2_route_set(const stmdev_ctx_t *ctx,
                                     i3g4250d_int2_route_t val);
int32_t i3g4250d_pin_int2_route_get(const stmdev_ctx_t *ctx,
                                     i3g4250d_int2_route_t *val);

typedef enum
{
  I3G4250D_PUSH_PULL    = 0,
  I3G4250D_OPEN_DRAIN   = 1,
} i3g4250d_pp_od_t;
int32_t i3g4250d_pin_mode_set(const stmdev_ctx_t *ctx,
                               i3g4250d_pp_od_t val);
int32_t i3g4250d_pin_mode_get(const stmdev_ctx_t *ctx,
                               i3g4250d_pp_od_t *val);

typedef enum
{
  I3G4250D_ACTIVE_HIGH   = 0,
  I3G4250D_ACTIVE_LOW    = 1,
} i3g4250d_h_lactive_t;
int32_t i3g4250d_pin_polarity_set(const stmdev_ctx_t *ctx,
                                  i3g4250d_h_lactive_t val);
int32_t i3g4250d_pin_polarity_get(const stmdev_ctx_t *ctx,
                                  i3g4250d_h_lactive_t *val);

typedef enum
{
  I3G4250D_INT_PULSED   = 0,
  I3G4250D_INT_LATCHED  = 1,
} i3g4250d_lir_t;
int32_t i3g4250d_int_notification_set(const stmdev_ctx_t *ctx,
                                      i3g4250d_lir_t val);
int32_t i3g4250d_int_notification_get(const stmdev_ctx_t *ctx,
                                      i3g4250d_lir_t *val);

int32_t i3g4250d_int_on_threshold_conf_set(const stmdev_ctx_t *ctx,
                                           i3g4250d_int1_cfg_t *val);
int32_t i3g4250d_int_on_threshold_conf_get(const stmdev_ctx_t *ctx,
                                           i3g4250d_int1_cfg_t *val);

typedef enum
{
  I3G4250D_INT1_ON_TH_AND  = 1,
  I3G4250D_INT1_ON_TH_OR   = 0,
} i3g4250d_and_or_t;
int32_t i3g4250d_int_on_threshold_mode_set(const stmdev_ctx_t *ctx,
                                           i3g4250d_and_or_t val);
int32_t i3g4250d_int_on_threshold_mode_get(const stmdev_ctx_t *ctx,
                                           i3g4250d_and_or_t *val);

int32_t i3g4250d_int_on_threshold_src_get(const stmdev_ctx_t *ctx,
                                          i3g4250d_int1_src_t *val);

int32_t i3g4250d_int_x_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
int32_t i3g4250d_int_x_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);

int32_t i3g4250d_int_y_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
int32_t i3g4250d_int_y_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);

int32_t i3g4250d_int_z_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
int32_t i3g4250d_int_z_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);

int32_t i3g4250d_int_on_threshold_dur_set(const stmdev_ctx_t *ctx,
                                          uint8_t val);
int32_t i3g4250d_int_on_threshold_dur_get(const stmdev_ctx_t *ctx,
                                          uint8_t *val);

int32_t i3g4250d_fifo_enable_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_fifo_enable_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_fifo_watermark_set(const stmdev_ctx_t *ctx, uint8_t val);
int32_t i3g4250d_fifo_watermark_get(const stmdev_ctx_t *ctx, uint8_t *val);

typedef enum
{
  I3G4250D_FIFO_BYPASS_MODE     = 0x00,
  I3G4250D_FIFO_MODE            = 0x01,
  I3G4250D_FIFO_STREAM_MODE     = 0x02,
} i3g4250d_fifo_mode_t;
int32_t i3g4250d_fifo_mode_set(const stmdev_ctx_t *ctx,
                               i3g4250d_fifo_mode_t val);
int32_t i3g4250d_fifo_mode_get(const stmdev_ctx_t *ctx,
                               i3g4250d_fifo_mode_t *val);

int32_t i3g4250d_fifo_data_level_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_fifo_empty_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_fifo_ovr_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);

int32_t i3g4250d_fifo_wtm_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);

#ifdef __cplusplus
}
#endif

#endif /* I3G4250D_REGS_H */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

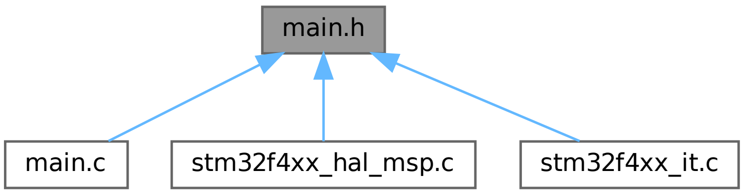
6.3 main.h File Reference

: Header for main.c file. This file contains the common defines of the application.

`#include "stm32f4xx_hal.h"`
Include dependency graph for main.h:



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



Macros

- #define GYRO_CS_OUT_GPIO_Port GPIOA
- #define GYRO_CS_OUT_Pin GPIO_PIN_4

Functions

- void Error_Handler (void)

6.3.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

6.3.2 Macro Definition Documentation

6.3.2.1 GYRO_CS_OUT_GPIO_Port

```
#define GYRO_CS_OUT_GPIO_Port GPIOA
```

6.3.2.2 GYRO_CS_OUT_Pin

```
#define GYRO_CS_OUT_Pin GPIO_PIN_4
```

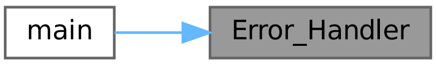
6.3.3 Function Documentation

6.3.3.1 Error_Handler()

```
void Error_Handler (
              void  )
```

Referenced by main().

Here is the caller graph for this function:



6.4 main.h

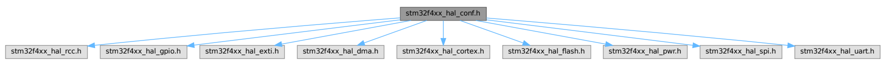
Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/
#define GYRO_CS_OUT_Pin GPIO_PIN_4
#define GYRO_CS_OUT_GPIO_Port GPIOA

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

6.5 stm32f4xx_hal_conf.h File Reference

```
#include "stm32f4xx_hal_rcc.h"
#include "stm32f4xx_hal_gpio.h"
#include "stm32f4xx_hal_exti.h"
#include "stm32f4xx_hal_dma.h"
#include "stm32f4xx_hal_cortex.h"
#include "stm32f4xx_hal_flash.h"
#include "stm32f4xx_hal_pwr.h"
#include "stm32f4xx_hal_spi.h"
#include "stm32f4xx_hal_uart.h"
```

Include dependency graph for stm32f4xx_hal_conf.h:



Macros

- #define assert_param(expr) ((void)0U)

  Include module's header file.
- #define DATA_CACHE_ENABLE 1U
- #define DP83848_PHY_ADDRESS
- #define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for receive ∗/
- #define ETH_RXBUFNB 4U /∗ 4 Rx buffers of size ETH_RX_BUF_SIZE ∗/
- #define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /∗ buffer size for transmit ∗/
- #define ETH_TXBUFNB 4U /∗ 4 Tx buffers of size ETH_TX_BUF_SIZE ∗/
- #define EXTERNAL_CLOCK_VALUE 12288000U

  External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.
- #define HAL_CORTEX_MODULE_ENABLED

- #define HAL_DMA_MODULE_ENABLED
- #define HAL_EXTI_MODULE_ENABLED
- #define HAL_FLASH_MODULE_ENABLED
- #define HAL_GPIO_MODULE_ENABLED
- #define HAL_MODULE_ENABLED

    This is the list of modules to be used in the HAL driver.
- #define HAL_PWR_MODULE_ENABLED
- #define HAL_RCC_MODULE_ENABLED
- #define HAL_SPI_MODULE_ENABLED
- #define HAL_UART_MODULE_ENABLED
- #define HSE_STARTUP_TIMEOUT 100U
- #define HSE_VALUE 25000000U

    Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).
- #define HSI_VALUE ((uint32_t)16000000U)

    Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).
- #define INSTRUCTION_CACHE_ENABLE 1U
- #define LSE_STARTUP_TIMEOUT 5000U
- #define LSE_VALUE 32768U

    External Low Speed oscillator (LSE) value.
- #define LSI_VALUE 32000U

    Internal Low Speed oscillator (LSI) value.
- #define MAC_ADDR0 2U

    Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.
- #define MAC_ADDR1 0U
- #define MAC_ADDR2 0U
- #define MAC_ADDR3 0U
- #define MAC_ADDR4 0U
- #define MAC_ADDR5 0U
- #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
- #define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
- #define PHY_BCR ((uint16_t)0x0000U)
- #define PHY_BSR ((uint16_t)0x0001U)
- #define PHY_CONFIG_DELAY 0x00000FFFU
- #define PHY_DUPLEX_STATUS ((uint16_t))
- #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
- #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
- #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
- #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
- #define PHY_ISOLATE ((uint16_t)0x0400U)
- #define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
- #define PHY_LINKED_STATUS ((uint16_t)0x0004U)
- #define PHY_LOOPBACK ((uint16_t)0x4000U)
- #define PHY_POWERDOWN ((uint16_t)0x0800U)
- #define PHY_READ_TO 0x0000FFFFU
- #define PHY_RESET ((uint16_t)0x8000U)
- #define PHY_RESET_DELAY 0x000000FFU
- #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
- #define PHY_SPEED_STATUS ((uint16_t))
- #define PHY_SR ((uint16_t))
- #define PHY_WRITE_TO 0x0000FFFFU
- #define PREFETCH_ENABLE 1U
- #define TICK_INT_PRIORITY 15U

- #define USE_HAL_ADC_REGISTER_CALLBACKS 0U /∗ ADC register callback disabled ∗/
- #define USE_HAL_CAN_REGISTER_CALLBACKS 0U /∗ CAN register callback disabled ∗/
- #define USE_HAL_CEC_REGISTER_CALLBACKS 0U /∗ CEC register callback disabled ∗/
- #define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /∗ CRYP register callback disabled ∗/
- #define USE_HAL_DAC_REGISTER_CALLBACKS 0U /∗ DAC register callback disabled ∗/
- #define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /∗ DCMI register callback disabled ∗/
- #define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /∗ DFSDM register callback disabled ∗/
- #define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /∗ DMA2D register callback disabled ∗/
- #define USE_HAL_DSI_REGISTER_CALLBACKS 0U /∗ DSI register callback disabled ∗/
- #define USE_HAL_ETH_REGISTER_CALLBACKS 0U /∗ ETH register callback disabled ∗/
- #define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /∗ FMPI2C register callback disabled ∗/
- #define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /∗ FMPSMBUS register callback disabled ∗/
- #define USE_HAL_HASH_REGISTER_CALLBACKS 0U /∗ HASH register callback disabled ∗/
- #define USE_HAL_HCD_REGISTER_CALLBACKS 0U /∗ HCD register callback disabled ∗/
- #define USE_HAL_I2C_REGISTER_CALLBACKS 0U /∗ I2C register callback disabled ∗/
- #define USE_HAL_I2S_REGISTER_CALLBACKS 0U /∗ I2S register callback disabled ∗/
- #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /∗ IRDA register callback disabled ∗/
- #define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /∗ LPTIM register callback disabled ∗/
- #define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /∗ LTDC register callback disabled ∗/
- #define USE_HAL_MMC_REGISTER_CALLBACKS 0U /∗ MMC register callback disabled ∗/
- #define USE_HAL_NAND_REGISTER_CALLBACKS 0U /∗ NAND register callback disabled ∗/
- #define USE_HAL_NOR_REGISTER_CALLBACKS 0U /∗ NOR register callback disabled ∗/
- #define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /∗ PCCARD register callback disabled ∗/
- #define USE_HAL_PCD_REGISTER_CALLBACKS 0U /∗ PCD register callback disabled ∗/
- #define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /∗ QSPI register callback disabled ∗/
- #define USE_HAL_RNG_REGISTER_CALLBACKS 0U /∗ RNG register callback disabled ∗/
- #define USE_HAL_RTC_REGISTER_CALLBACKS 0U /∗ RTC register callback disabled ∗/
- #define USE_HAL_SAI_REGISTER_CALLBACKS 0U /∗ SAI register callback disabled ∗/
- #define USE_HAL_SD_REGISTER_CALLBACKS 0U /∗ SD register callback disabled ∗/
- #define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /∗ SDRAM register callback disabled ∗/
- #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /∗ SMARTCARD register callback disabled ∗/
- #define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /∗ SMBUS register callback disabled ∗/
- #define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /∗ SPDIFRX register callback disabled ∗/
- #define USE_HAL_SPI_REGISTER_CALLBACKS 0U /∗ SPI register callback disabled ∗/
- #define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /∗ SRAM register callback disabled ∗/
- #define USE_HAL_TIM_REGISTER_CALLBACKS 0U /∗ TIM register callback disabled ∗/
- #define USE_HAL_UART_REGISTER_CALLBACKS 0U /∗ UART register callback disabled ∗/
- #define USE_HAL_USART_REGISTER_CALLBACKS 0U /∗ USART register callback disabled ∗/
- #define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /∗ WWDG register callback disabled ∗/
- #define USE_RTOS 0U
- #define USE_SPI_CRC 0U
- #define VDD_VALUE 3300U

  This is the HAL system configuration section.

6.5.1 Macro Definition Documentation

6.5.1.1 assert_param

```
#define assert_param(
            expr ) ((void)0U)
```

Include module's header file.

6.5.1.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 1U
```

6.5.1.3 DP83848_PHY_ADDRESS

```
#define DP83848_PHY_ADDRESS
```

6.5.1.4 ETH_RX_BUF_SIZE

```
#define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */
```

6.5.1.5 ETH_RXBUFNB

```
#define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
```

6.5.1.6 ETH_TX_BUF_SIZE

```
#define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */
```

6.5.1.7 ETH_TXBUFNB

```
#define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
```

6.5.1.8 EXTERNAL_CLOCK_VALUE

```
#define EXTERNAL_CLOCK_VALUE 12288000U
```

External clock source for I2S peripheral This value is used by the I2S HAL module to compute the I2S clock source frequency, this source is inserted directly through I2S_CKIN pad.

Value of the External audio frequency in Hz

6.5.1.9 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

6.5.1.10 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

6.5.1.11 HAL_EXTI_MODULE_ENABLED

```
#define HAL_EXTI_MODULE_ENABLED
```

6.5.1.12 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

6.5.1.13 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

6.5.1.14 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

6.5.1.15 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

6.5.1.16 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

6.5.1.17 HAL_SPI_MODULE_ENABLED

```
#define HAL_SPI_MODULE_ENABLED
```

6.5.1.18 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

6.5.1.19 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT 100U
```

Time out for HSE start up, in ms

6.5.1.20 HSE_VALUE

```
#define HSE_VALUE 25000000U
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

6.5.1.21 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)16000000U)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

6.5.1.22 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 1U
```

6.5.1.23 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT 5000U
```

Time out for LSE start up, in ms

6.5.1.24 LSE_VALUE

```
#define LSE_VALUE 32768U
```

External Low Speed oscillator (LSE) value.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature. Value of the External Low Speed oscillator in Hz

6.5.1.25 LSI_VALUE

```
#define LSI_VALUE 32000U
```

Internal Low Speed oscillator (LSI) value.

LSI Typical Value in Hz

6.5.1.26 MAC_ADDR0

```
#define MAC_ADDR0 2U
```

Uncomment the line below to expanse the "assert_param" macro in the HAL drivers code.

6.5.1.27 MAC_ADDR1

```
#define MAC_ADDR1 0U
```

6.5.1.28 MAC_ADDR2

```
#define MAC_ADDR2 0U
```

6.5.1.29 MAC_ADDR3

```
#define MAC_ADDR3 0U
```

6.5.1.30 MAC_ADDR4

```
#define MAC_ADDR4 0U
```

6.5.1.31 MAC_ADDR5

```
#define MAC_ADDR5 0U
```

6.5.1.32 PHY_AUTONEGO_COMPLETE

```
#define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
```

Auto-Negotiation process completed

6.5.1.33 PHY_AUTONEGOTIATION

```
#define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
```

Enable auto-negotiation function

6.5.1.34 PHY_BCR

```
#define PHY_BCR ((uint16_t)0x0000U)
```

Transceiver Basic Control Register

6.5.1.35 PHY_BSR

```
#define PHY_BSR ((uint16_t)0x0001U)
```

Transceiver Basic Status Register

6.5.1.36 PHY_CONFIG_DELAY

```
#define PHY_CONFIG_DELAY 0x00000FFFU
```

6.5.1.37 PHY_DUPLEX_STATUS

```
#define PHY_DUPLEX_STATUS ((uint16_t))
```

PHY Duplex mask

6.5.1.38 PHY_FULLDUPLEX_100M

```
#define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
```

Set the full-duplex mode at 100 Mb/s

6.5.1.39 PHY_FULLDUPLEX_10M

```
#define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
```

Set the full-duplex mode at 10 Mb/s

6.5.1.40 PHY_HALFDUPLEX_100M

```
#define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
```

Set the half-duplex mode at 100 Mb/s

6.5.1.41 PHY_HALFDUPLEX_10M

```
#define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
```

Set the half-duplex mode at 10 Mb/s

6.5.1.42 PHY_ISOLATE

```
#define PHY_ISOLATE ((uint16_t)0x0400U)
```

Isolate PHY from MII

6.5.1.43 PHY_JABBER_DETECTION

```
#define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
```

Jabber condition detected

6.5.1.44 PHY_LINKED_STATUS

```
#define PHY_LINKED_STATUS ((uint16_t)0x0004U)
```

Valid link established

6.5.1.45 PHY_LOOPBACK

```
#define PHY_LOOPBACK ((uint16_t)0x4000U)
```

Select loop-back mode

6.5.1.46 PHY_POWERDOWN

```
#define PHY_POWERDOWN ((uint16_t)0x0800U)
```

Select the power down mode

6.5.1.47 PHY_READ_TO

```
#define PHY_READ_TO 0x0000FFFFU
```

6.5.1.48 PHY_RESET

```
#define PHY_RESET ((uint16_t)0x8000U)
```

PHY Reset

6.5.1.49 PHY_RESET_DELAY

```
#define PHY_RESET_DELAY 0x000000FFU
```

6.5.1.50 PHY_RESTART_AUTONEGOTIATION

```
#define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
```

Restart auto-negotiation function

6.5.1.51 PHY_SPEED_STATUS

```
#define PHY_SPEED_STATUS ((uint16_t))
```

PHY Speed mask

6.5.1.52 PHY_SR

```
#define PHY_SR ((uint16_t))
```

PHY status register Offset

6.5.1.53 PHY_WRITE_TO

```
#define PHY_WRITE_TO 0x0000FFFFU
```

6.5.1.54 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1U
```

6.5.1.55 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY 15U
```

tick interrupt priority

6.5.1.56 USE_HAL_ADC_REGISTER_CALLBACKS

```
#define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
```

6.5.1.57 USE_HAL_CAN_REGISTER_CALLBACKS

```
#define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
```

6.5.1.58 USE_HAL_CEC_REGISTER_CALLBACKS

```
#define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
```

6.5.1.59 USE_HAL_CRYP_REGISTER_CALLBACKS

```
#define USE_HAL_CRYP_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */
```

6.5.1.60 USE_HAL_DAC_REGISTER_CALLBACKS

```
#define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
```

6.5.1.61 USE_HAL_DCMI_REGISTER_CALLBACKS

```
#define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */
```

6.5.1.62 USE_HAL_DFSDM_REGISTER_CALLBACKS

```
#define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */
```

6.5.1.63 USE_HAL_DMA2D_REGISTER_CALLBACKS

```
#define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */
```

6.5.1.64 USE_HAL_DSI_REGISTER_CALLBACKS

```
#define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */
```

6.5.1.65 USE_HAL_ETH_REGISTER_CALLBACKS

```
#define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
```

6.5.1.66 USE_HAL_FMPI2C_REGISTER_CALLBACKS

```
#define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */
```

6.5.1.67 USE_HAL_FMPSMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_FMPSMBUS_REGISTER_CALLBACKS 0U /* FMPSMBUS register callback disabled */
```

6.5.1.68 USE_HAL_HASH_REGISTER_CALLBACKS

```
#define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */
```

6.5.1.69 USE_HAL_HCD_REGISTER_CALLBACKS

```
#define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
```

6.5.1.70 USE_HAL_I2C_REGISTER_CALLBACKS

```
#define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
```

6.5.1.71 USE_HAL_I2S_REGISTER_CALLBACKS

```
#define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
```

6.5.1.72 USE_HAL_IRDA_REGISTER_CALLBACKS

```
#define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
```

6.5.1.73 USE_HAL_LPTIM_REGISTER_CALLBACKS

```
#define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */
```

6.5.1.74 USE_HAL_LTDC_REGISTER_CALLBACKS

```
#define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */
```

6.5.1.75 USE_HAL_MMC_REGISTER_CALLBACKS

```
#define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
```

6.5.1.76 USE_HAL_NAND_REGISTER_CALLBACKS

```
#define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
```

6.5.1.77 USE_HAL_NOR_REGISTER_CALLBACKS

```
#define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
```

6.5.1.78 USE_HAL_PCCARD_REGISTER_CALLBACKS

```
#define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
```

6.5.1.79 USE_HAL_PCD_REGISTER_CALLBACKS

```
#define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
```

6.5.1.80 USE_HAL_QSPI_REGISTER_CALLBACKS

```
#define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */
```

6.5.1.81 USE_HAL_RNG_REGISTER_CALLBACKS

```
#define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */
```

6.5.1.82 USE_HAL_RTC_REGISTER_CALLBACKS

```
#define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
```

6.5.1.83 USE_HAL_SAI_REGISTER_CALLBACKS

```
#define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */
```

6.5.1.84 USE_HAL_SD_REGISTER_CALLBACKS

```
#define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
```

6.5.1.85 USE_HAL_SDRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */
```

6.5.1.86 USE_HAL_SMARTCARD_REGISTER_CALLBACKS

```
#define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
```

6.5.1.87 USE_HAL_SMBUS_REGISTER_CALLBACKS

```
#define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */
```

6.5.1.88 USE_HAL_SPDIFRX_REGISTER_CALLBACKS

```
#define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
```

6.5.1.89 USE_HAL_SPI_REGISTER_CALLBACKS

```
#define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
```

6.5.1.90 USE_HAL_SRAM_REGISTER_CALLBACKS

```
#define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
```

6.5.1.91 USE_HAL_TIM_REGISTER_CALLBACKS

```
#define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
```

6.5.1.92 USE_HAL_UART_REGISTER_CALLBACKS

```
#define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
```

6.5.1.93 USE_HAL_USART_REGISTER_CALLBACKS

```
#define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
```

6.5.1.94 USE_HAL_WWDG_REGISTER_CALLBACKS

```
#define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
```

6.5.1.95 USE_RTOS

```
#define USE_RTOS 0U
```

6.5.1.96 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

6.5.1.97 VDD_VALUE

```
#define VDD_VALUE 3300U
```

This is the HAL system configuration section.

Value of VDD in mv

6.6 stm32f4xx_hal_conf.h

Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
#define HAL_MODULE_ENABLED

  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
/* #define HAL_I2C_MODULE_ENABLED */
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
#define HAL_SPI_MODULE_ENABLED
/* #define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
```

```
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED

/* ########################## HSE/HSI Values adaptation ##################### */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    25000000U
#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U
#endif /* HSE_STARTUP_TIMEOUT */

#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U)
#endif /* HSI_VALUE */

#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U
#endif /* LSI_VALUE */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U
#endif /* LSE_VALUE */

#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U
#endif /* LSE_STARTUP_TIMEOUT */

#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U
#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different HSE,
   ===   you can define the HSE value in your toolchain compiler preprocessor. */

/* ########################### System Configuration ######################### */
#define  VDD_VALUE            3300U
#define  TICK_INT_PRIORITY            15U
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback disabled       */
#define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback disabled       */
#define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback disabled      */
#define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback disabled       */
#define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback disabled       */
#define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback disabled    */
#define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback disabled  */
#define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback disabled       */
#define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback disabled      */
#define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback disabled     */
#define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback disabled      */
#define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback disabled       */
#define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback disabled      */
#define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback disabled       */
#define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback disabled    */
#define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback disabled       */
#define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback disabled      */
#define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback disabled       */
#define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback disabled       */
#define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback disabled       */
#define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback disabled        */
#define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register callback disabled */
#define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback disabled     */
#define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback disabled      */
#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback disabled   */
#define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback disabled     */
#define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback disabled       */
#define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback disabled       */
#define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback disabled      */
#define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback disabled     */
#define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback disabled      */

/* ########################## Assert Selection ############################## */
/* #define USE_FULL_ASSERT    1U */

/* ################## Ethernet peripheral configuration ##################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for receive               */
#define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for transmit              */
#define ETH_RXBUFNB                    4U       /* 4 Rx buffers of size ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY                 0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY                0x00000FFFU

#define PHY_READ_TO                     0x0000FFFFU
#define PHY_WRITE_TO                    0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                         ((uint16_t)0x0000U)
#define PHY_BSR                         ((uint16_t)0x0001U)
#define PHY_RESET                       ((uint16_t)0x8000U)
#define PHY_LOOPBACK                    ((uint16_t)0x4000U)
#define PHY_FULLDUPLEX_100M             ((uint16_t)0x2100U)
#define PHY_HALFDUPLEX_100M             ((uint16_t)0x2000U)
#define PHY_FULLDUPLEX_10M              ((uint16_t)0x0100U)
#define PHY_HALFDUPLEX_10M              ((uint16_t)0x0000U)
#define PHY_AUTONEGOTIATION             ((uint16_t)0x1000U)
#define PHY_RESTART_AUTONEGOTIATION     ((uint16_t)0x0200U)
#define PHY_POWERDOWN                   ((uint16_t)0x0800U)
#define PHY_ISOLATE                     ((uint16_t)0x0400U)
#define PHY_AUTONEGO_COMPLETE           ((uint16_t)0x0020U)
#define PHY_LINKED_STATUS               ((uint16_t)0x0004U)
#define PHY_JABBER_DETECTION            ((uint16_t)0x0002U)
/* Section 4: Extended PHY Registers */
#define PHY_SR                          ((uint16_t))
#define PHY_SPEED_STATUS                ((uint16_t))
#define PHY_DUPLEX_STATUS               ((uint16_t))
/* ################## SPI peripheral configuration ########################## */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/

#define USE_SPI_CRC                     0U

/* Includes ------------------------------------------------------------------*/
#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */

#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */

#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */

#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */

#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */

```

```
#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */

#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */

#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */

#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
#endif /* HAL_DMA2D_MODULE_ENABLED */

#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */

#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */

#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */

#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */

#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */

#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */

#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */

#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */

#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */

#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */

#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */

#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */

#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */

#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */

#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */

#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */

#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */
```

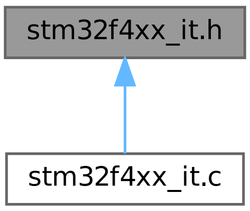
```

#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */

#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */

#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */

#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */

#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */

#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */

#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */

#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */

#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */

#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */

#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */

#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */

#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */

#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */

#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */

#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */

#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */

#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */

#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */

#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */

/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
```

```
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */

#ifdef __cplusplus
}
#endif

#endif /* __STM32F4xx_HAL_CONF_H */
```

6.7 stm32f4xx_it.h File Reference

This file contains the headers of the interrupt handlers.

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



Functions

- void BusFault_Handler (void)

  This function handles Pre-fetch fault, memory access fault.
- void DebugMon_Handler (void)

  This function handles Debug monitor.
- void HardFault_Handler (void)

  This function handles Hard fault interrupt.
- void MemManage_Handler (void)

  This function handles Memory management fault.
- void NMI_Handler (void)

  This function handles Non maskable interrupt.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SVC_Handler (void)

  This function handles System service call via SWI instruction.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void UsageFault_Handler (void)

  This function handles Undefined instruction or illegal state.

6.7.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

6.7.2 Function Documentation

6.7.2.1 BusFault_Handler()

```
void BusFault_Handler (
             void  )
```

This function handles Pre-fetch fault, memory access fault.

6.7.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
             void  )
```

This function handles Debug monitor.

6.7.2.3 HardFault_Handler()

```
void HardFault_Handler (
             void  )
```

This function handles Hard fault interrupt.

6.7.2.4 MemManage_Handler()

```
void MemManage_Handler (
             void  )
```

This function handles Memory management fault.

6.7.2.5 NMI_Handler()

```
void NMI_Handler (
             void  )
```

This function handles Non maskable interrupt.

6.7.2.6 PendSV_Handler()

```
void PendSV_Handler (
            void  )
```

This function handles Pendable request for system service.

6.7.2.7 SVC_Handler()

```
void SVC_Handler (
            void  )
```

This function handles System service call via SWI instruction.

6.7.2.8 SysTick_Handler()

```
void SysTick_Handler (
            void  )
```

This function handles System tick timer.

6.7.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
            void  )
```

This function handles Undefined instruction or illegal state.

6.8 stm32f4xx_it.h

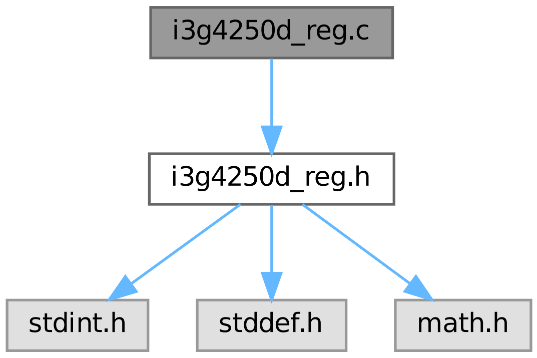
Go to the documentation of this file.

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

#ifdef __cplusplus
}
#endif

#endif /* __STM32F4xx_IT_H */
```

6.9 i3g4250d_reg.c File Reference

I3G4250D driver file.

`#include "i3g4250d_reg.h"`
Include dependency graph for i3g4250d_reg.c:



Functions

- int32_t i3g4250d_angular_rate_raw_get (const stmdev_ctx_t ∗ctx, int16_t ∗val)

  Angular rate sensor. The value is expressed as a 16-bit word in two's complement.[get].
- int32_t i3g4250d_boot_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Reboot memory content. Reload the calibration parameters.[get].
- int32_t i3g4250d_boot_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  Reboot memory content. Reload the calibration parameters.[set].
- int32_t i3g4250d_data_format_get (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t ∗val)

  Big/Little Endian data selection.[get].
- int32_t i3g4250d_data_format_set (const stmdev_ctx_t ∗ctx, i3g4250d_ble_t val)

Big/Little Endian data selection.[set].
- int32_t i3g4250d_data_rate_get (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t ∗val)

  Accelerometer data rate selection.[get].
- int32_t i3g4250d_data_rate_set (const stmdev_ctx_t ∗ctx, i3g4250d_dr_t val)

  Accelerometer data rate selection.[set].
- int32_t i3g4250d_device_id_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Device Who amI.[get].
- int32_t i3g4250d_fifo_data_level_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO stored data level[get].
- int32_t i3g4250d_fifo_empty_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOemptybit.[get].
- int32_t i3g4250d_fifo_enable_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFOenable.[get].
- int32_t i3g4250d_fifo_enable_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFOenable.[set].
- int32_t i3g4250d_fifo_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t ∗val)

  FIFO mode selection.[get].
- int32_t i3g4250d_fifo_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_fifo_mode_t val)

  FIFO mode selection.[set].
- int32_t i3g4250d_fifo_ovr_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Overrun bit status.[get].
- int32_t i3g4250d_fifo_watermark_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  FIFO watermark level selection.[get].
- int32_t i3g4250d_fifo_watermark_set (const stmdev_ctx_t ∗ctx, uint8_t val)

  FIFO watermark level selection.[set].
- int32_t i3g4250d_fifo_wtm_flag_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Watermark status:[get] 0: FIFO filling is lower than WTM level; 1: FIFO filling is equal or higher than WTM level)
- int32_t i3g4250d_filter_path_get (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t ∗val)

  Out/FIFO selection path. [get].
- int32_t i3g4250d_filter_path_internal_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t ∗val)

  Interrupt generator selection path.[get].
- int32_t i3g4250d_filter_path_internal_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_sel_t val)

  Interrupt generator selection path.[set].
- int32_t i3g4250d_filter_path_set (const stmdev_ctx_t ∗ctx, i3g4250d_out_sel_t val)

  Out/FIFO selection path. [set].
- int32_t i3g4250d_flag_data_ready_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

  Accelerometer new data available.[get].
- float_t i3g4250d_from_fs245dps_to_mdps (int16_t lsb)
- float_t i3g4250d_from_lsb_to_celsius (int16_t lsb)
- int32_t i3g4250d_full_scale_get (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t ∗val)

  Gyroscope full-scale selection.[get].
- int32_t i3g4250d_full_scale_set (const stmdev_ctx_t ∗ctx, i3g4250d_fs_t val)

  Gyroscope full-scale selection.[set].
- int32_t i3g4250d_hp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t ∗val)

  High-pass filter bandwidth selection.[get].
- int32_t i3g4250d_hp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpcf_t val)

  High-pass filter bandwidth selection.[set].
- int32_t i3g4250d_hp_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t ∗val)

  High-pass filter mode selection. [get].
- int32_t i3g4250d_hp_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_hpm_t val)

  High-pass filter mode selection. [set].

- int32_t i3g4250d_hp_reference_value_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

    Reference value for high-pass filter.[get].
- int32_t i3g4250d_hp_reference_value_set (const stmdev_ctx_t ∗ctx, uint8_t val)

    Reference value for high-pass filter.[set].
- int32_t i3g4250d_int_notification_get (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t ∗val)

    Latched/pulsed interrupt.[get].
- int32_t i3g4250d_int_notification_set (const stmdev_ctx_t ∗ctx, i3g4250d_lir_t val)

    Latched/pulsed interrupt.[set].
- int32_t i3g4250d_int_on_threshold_conf_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

    Configure the interrupt threshold sign.[get].
- int32_t i3g4250d_int_on_threshold_conf_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_cfg_t ∗val)

    Configure the interrupt threshold sign.[set].
- int32_t i3g4250d_int_on_threshold_dur_get (const stmdev_ctx_t ∗ctx, uint8_t ∗val)

    Durationvalue.[get].
- int32_t i3g4250d_int_on_threshold_dur_set (const stmdev_ctx_t ∗ctx, uint8_t val)

    Durationvalue.[set].
- int32_t i3g4250d_int_on_threshold_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t ∗val)

    AND/OR combination of interrupt events.[get].
- int32_t i3g4250d_int_on_threshold_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_and_or_t val)

    AND/OR combination of interrupt events.[set].
- int32_t i3g4250d_int_on_threshold_src_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_src_t ∗val)

    int_on_threshold_src: [get]
- int32_t i3g4250d_int_x_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

    Interrupt threshold on X.[get].
- int32_t i3g4250d_int_x_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

    Interrupt threshold on X.[set].
- int32_t i3g4250d_int_y_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

    Interrupt threshold on Y.[get].
- int32_t i3g4250d_int_y_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

    Interrupt threshold on Y.[set].
- int32_t i3g4250d_int_z_threshold_get (const stmdev_ctx_t ∗ctx, uint16_t ∗val)

    Interrupt threshold on Z.[get].
- int32_t i3g4250d_int_z_threshold_set (const stmdev_ctx_t ∗ctx, uint16_t val)

    Interrupt threshold on Z.[set].
- int32_t i3g4250d_lp_bandwidth_get (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t ∗val)

    Lowpass filter bandwidth selection.[get].
- int32_t i3g4250d_lp_bandwidth_set (const stmdev_ctx_t ∗ctx, i3g4250d_bw_t val)

    Lowpass filter bandwidth selection.[set].
- int32_t i3g4250d_pin_int1_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t ∗val)

    Select the signal that need to route on int1 pad.[get].
- int32_t i3g4250d_pin_int1_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int1_route_t val)

    Select the signal that need to route on int1 pad.[set].
- int32_t i3g4250d_pin_int2_route_get (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t ∗val)

    Select the signal that need to route on int2 pad.[get].
- int32_t i3g4250d_pin_int2_route_set (const stmdev_ctx_t ∗ctx, i3g4250d_int2_route_t val)

    Select the signal that need to route on int2 pad.[set].
- int32_t i3g4250d_pin_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t ∗val)

    Push-pull/open drain selection on interrupt pads.[get].
- int32_t i3g4250d_pin_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_pp_od_t val)

    Push-pull/open drain selection on interrupt pads.[set].
- int32_t i3g4250d_pin_polarity_get (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t ∗val)

Pin active-high/low.[get].
- int32_t i3g4250d_pin_polarity_set (const stmdev_ctx_t ∗ctx, i3g4250d_h_lactive_t val)

  Pin active-high/low.[set].
- int32_t __weak i3g4250d_read_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Read generic device register.
- int32_t i3g4250d_self_test_get (const stmdev_ctx_t ∗ctx, i3g4250d_st_t ∗val)

  Angular rate sensor self-test enable. [get].
- int32_t i3g4250d_self_test_set (const stmdev_ctx_t ∗ctx, i3g4250d_st_t val)

  Angular rate sensor self-test enable. [set].
- int32_t i3g4250d_spi_mode_get (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t ∗val)

  SPI Serial Interface Mode selection.[get].
- int32_t i3g4250d_spi_mode_set (const stmdev_ctx_t ∗ctx, i3g4250d_sim_t val)

  SPI Serial Interface Mode selection.[set].
- int32_t i3g4250d_status_reg_get (const stmdev_ctx_t ∗ctx, i3g4250d_status_reg_t ∗val)

  The STATUS_REG register is read by the primary interface.[get].
- int32_t i3g4250d_temperature_raw_get (const stmdev_ctx_t ∗ctx, uint8_t ∗buff)

  Temperature data.[get].
- int32_t __weak i3g4250d_write_reg (const stmdev_ctx_t ∗ctx, uint8_t reg, uint8_t ∗data, uint16_t len)

  Write generic device register.

6.9.1 Detailed Description

I3G4250D driver file.

Author

Sensors Software Solution Team

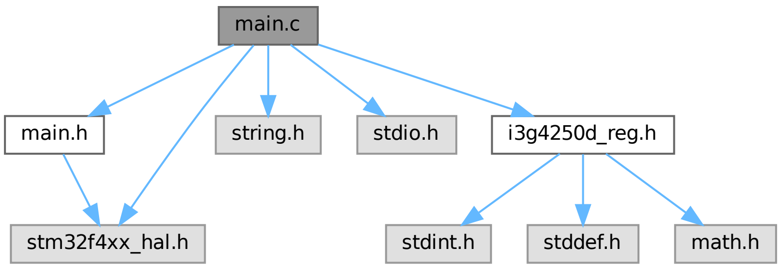
Attention

6.10 main.c File Reference

: Main program body

```
#include "main.h"
#include <string.h>
#include <stdio.h>
#include "i3g4250d_reg.h"
#include "stm32f4xx_hal.h"
```

Include dependency graph for main.c:



Macros

- #define BOOT_TIME 10

Functions

- int main (void)

  The application entry point.
- void SystemClock_Config (void)

Variables

- SPI_HandleTypeDef hspi1
- UART_HandleTypeDef huart2

6.10.1 Detailed Description

: Main program body

Attention

6.10.2 Macro Definition Documentation

6.10.2.1 BOOT_TIME

```
#define BOOT_TIME 10
```

6.10.3 Function Documentation

6.10.3.1 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

System Clock Configuration

Return values

None	

Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef structure.

Activate the Over-Drive mode

Initializes the CPU, AHB and APB buses clocks

SPI1 Initialization Function

Parameters

None	

Return values

None	

USART2 Initialization Function

Parameters

None	

Return values

None	

GPIO Initialization Function

Parameters

None	

Return values

None	

Writes data to a specified register of the Gyro.

Sends register address followed by the data to be written. The SPI write operation is executed by setting RW bit to `0` as per the datasheet.

Parameters

`in`	*handle*	SPI handle to use for communication.
`in`	*reg*	Register address to which data needs to be written.
`in`	*bufp*	Pointer to the data buffer to be written.
`in`	*len*	Length of the data buffer in bytes.

Returns

int32_t 0 on success, error code otherwise.

Reads data from a specified register of the Gyro device.

This function communicates with the device over SPI, sending the register address and then receiving the data from the device. The SPI read operation is executed by setting the RW bit to `1` as per the datasheet. The CS pin is used to select the device, and SPI transactions are done with the SPI handle.

Parameters

`in`	*handle*	SPI handle to use for communication.
`in`	*reg*	Register address from which data needs to be read.
`out`	*bufp*	Pointer to the buffer where the read data will be stored.
`in`	*len*	Length of the data buffer in bytes to be read.

Returns

int32_t 0 on success, error code otherwise.

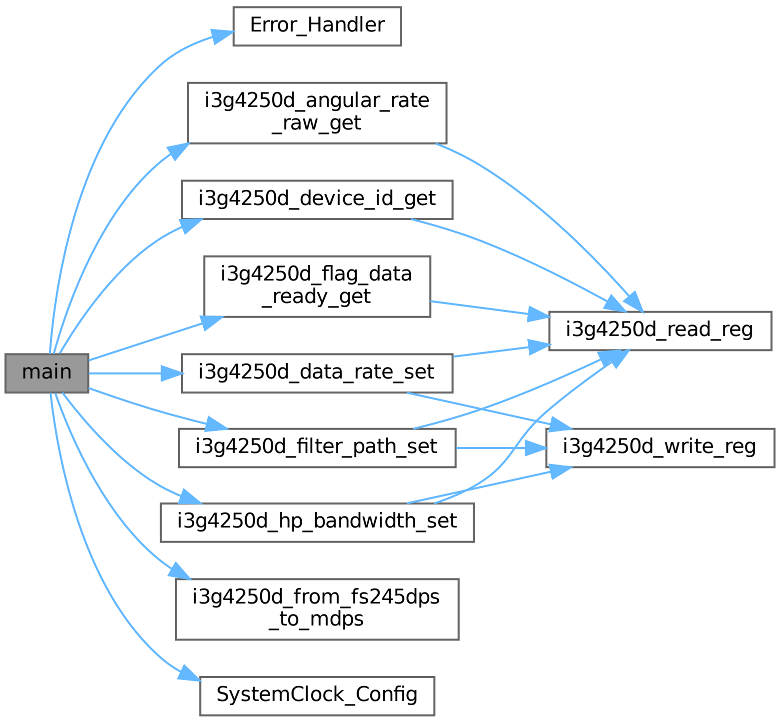
This function is executed in case of error occurrence.

Return values

None	

References BOOT_TIME, Error_Handler(), GYRO_CS_OUT_GPIO_Port, GYRO_CS_OUT_Pin, stmdev_ctx_t::handle, hspi1, huart2, i3g4250d_angular_rate_raw_get(), i3g4250d_data_rate_set(), i3g4250d_device_id_get(), i3g4250d_filter_path_set(), i3g4250d_flag_data_ready_get(), i3g4250d_from_fs245dps_to_mdps(), i3g4250d_hp_bandwidth_set(), I3G4250D_HP_LEVEL_3, I3G4250D_ID, I3G4250D_LPF1_HP_ON_OUT, I3G4250D_ODR_100Hz, stmdev_ctx_t::mdelay, stmdev_ctx_t::read_reg, SystemClock_Config(), and stmdev_ctx_t::write_reg.

Here is the call graph for this function:

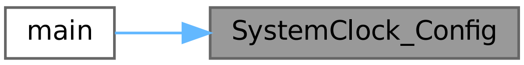


6.10.3.2 SystemClock_Config()

```
void SystemClock_Config (
            void  )
```

Referenced by main().

Here is the caller graph for this function:



6.10.4 Variable Documentation

6.10.4.1 hspi1

```
SPI_HandleTypeDef hspi1
```

Referenced by main().

6.10.4.2 huart2

```
UART_HandleTypeDef huart2
```

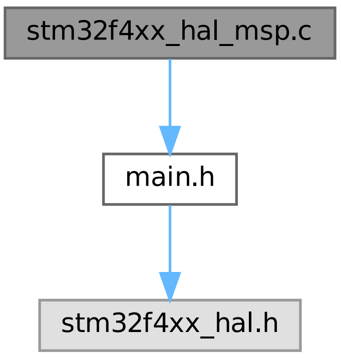
Referenced by main().

6.11 stm32f4xx_hal_msp.c File Reference

This file provides code for the MSP Initialization and de-Initialization codes.

```
#include "main.h"
```

Include dependency graph for stm32f4xx_hal_msp.c:



Functions

- void HAL_MspInit (void)
- void HAL_SPI_MspDeInit (SPI_HandleTypeDef ∗hspi)

  SPI MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_SPI_MspInit (SPI_HandleTypeDef ∗hspi)

  SPI MSP Initialization This function configures the hardware resources used in this example.
- void HAL_UART_MspDeInit (UART_HandleTypeDef ∗huart)

  UART MSP De-Initialization This function freeze the hardware resources used in this example.
- void HAL_UART_MspInit (UART_HandleTypeDef ∗huart)

  UART MSP Initialization This function configures the hardware resources used in this example.

6.11.1 Detailed Description

This file provides code for the MSP Initialization and de-Initialization codes.

Attention

6.11.2 Function Documentation

6.11.2.1 HAL_MspInit()

```
void HAL_MspInit (
            void )
```

Initializes the Global MSP.

6.11.2.2 HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (
            SPI_HandleTypeDef * hspi )
```

SPI MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

hspi	SPI handle pointer

Return values

None	

SPI1 GPIO Configuration PA5 ------> SPI1_SCK PA6 ------> SPI1_MISO PA7 ------> SPI1_MOSI

6.11.2.3 HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (
            SPI_HandleTypeDef * hspi )
```

SPI MSP Initialization This function configures the hardware resources used in this example.

Parameters

hspi	SPI handle pointer

Return values

None	

SPI1 GPIO Configuration PA5 ------> SPI1_SCK PA6 ------> SPI1_MISO PA7 ------> SPI1_MOSI

6.11.2.4 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
            UART_HandleTypeDef * huart )
```

UART MSP De-Initialization This function freeze the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

USART2 GPIO Configuration PA2 ------> USART2_TX PA3 ------> USART2_RX

6.11.2.5 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
            UART_HandleTypeDef * huart )
```

UART MSP Initialization This function configures the hardware resources used in this example.

Parameters

huart	UART handle pointer

Return values

None	

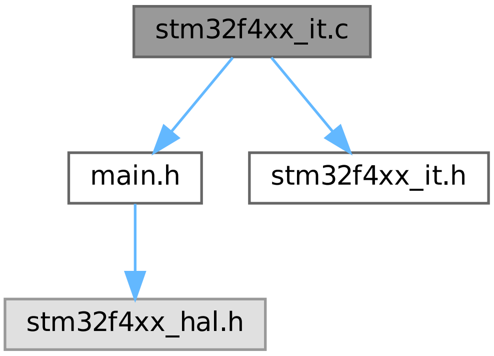
USART2 GPIO Configuration PA2 ----> USART2_TX PA3 ----> USART2_RX

6.12 stm32f4xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_it.h"
```

Include dependency graph for stm32f4xx_it.c:



Functions

- void BusFault_Handler (void)

  This function handles Pre-fetch fault, memory access fault.
- void DebugMon_Handler (void)

  This function handles Debug monitor.
- void HardFault_Handler (void)

  This function handles Hard fault interrupt.
- void MemManage_Handler (void)

  This function handles Memory management fault.
- void NMI_Handler (void)

  This function handles Non maskable interrupt.
- void PendSV_Handler (void)

  This function handles Pendable request for system service.
- void SVC_Handler (void)

  This function handles System service call via SWI instruction.
- void SysTick_Handler (void)

  This function handles System tick timer.
- void UsageFault_Handler (void)

  This function handles Undefined instruction or illegal state.

6.12.1 Detailed Description

Interrupt Service Routines.

Attention

6.12.2 Function Documentation

6.12.2.1 BusFault_Handler()

```
void BusFault_Handler (
            void  )
```

This function handles Pre-fetch fault, memory access fault.

6.12.2.2 DebugMon_Handler()

```
void DebugMon_Handler (
            void  )
```

This function handles Debug monitor.

6.12.2.3 HardFault_Handler()

```
void HardFault_Handler (
            void  )
```

This function handles Hard fault interrupt.

6.12.2.4 MemManage_Handler()

```
void MemManage_Handler (
            void  )
```

This function handles Memory management fault.

6.12.2.5 NMI_Handler()

```
void NMI_Handler (
            void  )
```

This function handles Non maskable interrupt.

6.12.2.6 PendSV_Handler()

```
void PendSV_Handler (
            void )
```

This function handles Pendable request for system service.

6.12.2.7 SVC_Handler()

```
void SVC_Handler (
            void )
```

This function handles System service call via SWI instruction.

6.12.2.8 SysTick_Handler()

```
void SysTick_Handler (
            void )
```

This function handles System tick timer.

6.12.2.9 UsageFault_Handler()

```
void UsageFault_Handler (
            void )
```

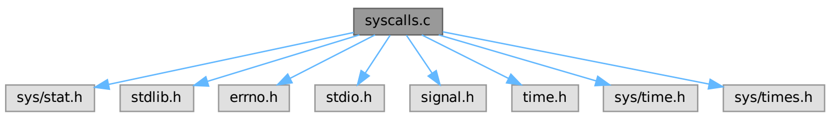
This function handles Undefined instruction or illegal state.

6.13 syscalls.c File Reference

STM32CubeIDE Minimal System calls file.

```
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>
```

Include dependency graph for syscalls.c:



Functions

- __attribute__ ((weak))
- int __io_getchar (void)
- int __io_putchar (int ch) __attribute__((weak))
- int _close (int file)
- int _execve (char ∗name, char ∗∗argv, char ∗∗env)
- void _exit (int status)
- int _fork (void)
- int _fstat (int file, struct stat ∗st)
- int _getpid (void)
- int _isatty (int file)
- int _kill (int pid, int sig)
- int _link (char ∗old, char ∗new)
- int _lseek (int file, int ptr, int dir)
- int _open (char ∗path, int flags,...)
- int _stat (char ∗file, struct stat ∗st)
- int _times (struct tms ∗buf)
- int _unlink (char ∗name)
- int _wait (int ∗status)
- void initialise_monitor_handles ()

Variables

- char ∗∗ environ = __env

6.13.1 Detailed Description

STM32CubeIDE Minimal System calls file.

Author

Auto-generated by STM32CubeIDE

```
For more information about which c-functions
need which of these lowlevel functions
please consult the Newlib libc-manual
```

Attention

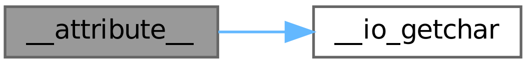
6.13.2 Function Documentation

6.13.2.1 __attribute__()

```
__attribute__ (
              (weak) )
```

References __io_getchar().

Here is the call graph for this function:

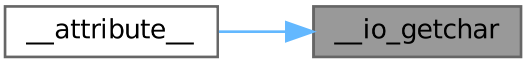


6.13.2.2 __io_getchar()

```
int __io_getchar (
              void ) [extern]
```

Referenced by __attribute__().

Here is the caller graph for this function:



6.13.2.3 __io_putchar()

```
int __io_putchar (
              int ch ) [extern]
```

6.13.2.4 _close()

```
int _close (
              int file )
```

6.13.2.5 _execve()

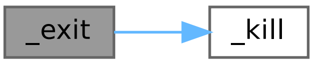
```
int _execve (
            char * name,
            char ** argv,
            char ** env )
```

6.13.2.6 _exit()

```
void _exit (
            int status )
```

References _kill().

Here is the call graph for this function:



6.13.2.7 _fork()

```
int _fork (
            void  )
```

6.13.2.8 _fstat()

```
int _fstat (
            int file,
            struct stat * st )
```

6.13.2.9 _getpid()

```
int _getpid (
            void  )
```

6.13.2.10 _isatty()

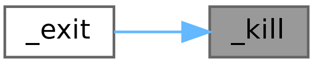
```
int _isatty (
            int file )
```

6.13.2.11 _kill()

```
int _kill (
             int pid,
             int sig )
```

Referenced by _exit().

Here is the caller graph for this function:



6.13.2.12 _link()

```
int _link (
             char * old,
             char * new )
```

6.13.2.13 _lseek()

```
int _lseek (
             int file,
             int ptr,
             int dir )
```

6.13.2.14 _open()

```
int _open (
             char * path,
             int flags,
              ...  )
```

6.13.2.15 _stat()

```
int _stat (
             char * file,
             struct stat * st )
```

6.13.2.16 _times()

```
int _times (
             struct tms * buf )
```

6.13.2.17 _unlink()

```
int _unlink (
            char * name )
```

6.13.2.18 _wait()

```
int _wait (
            int * status )
```

6.13.2.19 initialise_monitor_handles()

```
void initialise_monitor_handles ( )
```

6.13.3 Variable Documentation

6.13.3.1 environ

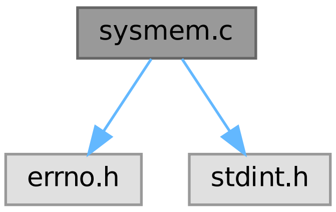
```
char** environ = __env
```

6.14 sysmem.c File Reference

STM32CubeIDE System Memory calls file.

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

Include dependency graph for sysmem.c:



Functions

- void ∗ _sbrk (ptrdiff_t incr)

    _sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

6.14.1 Detailed Description

STM32CubeIDE System Memory calls file.

Author

Generated by STM32CubeIDE

```
For more information about which C functions
need which of these lowlevel functions
please consult the newlib libc manual
```

Attention

6.14.2 Function Documentation

6.14.2.1 _sbrk()

```
void * _sbrk (
             ptrdiff_t incr )
```

_sbrk() allocates memory to the newlib heap and is used by malloc and others from the C library

```
* ############################################################################
* #   .data  #  .bss  #       newlib heap       #          MSP stack          #
* #          #        #                         # Reserved by _Min_Stack_Size #
* ############################################################################
* ^-- RAM start      ^-- _end                             _estack, RAM end --^
*
```

This implementation starts allocating at the '_end' linker symbol The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack The implementation considers '_estack' linker symbol to be RAM end NOTE: If the MSP stack, at any point during execution, grows larger than the reserved size, please increase the '_Min_Stack_Size'.

Parameters

incr	Memory size

Returns

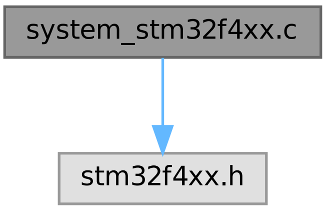
Pointer to allocated memory

6.15 system_stm32f4xx.c File Reference

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

```
#include "stm32f4xx.h"
```

Include dependency graph for system_stm32f4xx.c:



Macros

- #define HSE_VALUE ((uint32_t)25000000)
- #define HSI_VALUE ((uint32_t)16000000)

Functions

- void SystemCoreClockUpdate (void)

  Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- void SystemInit (void)

  Setup the microcontroller system Initialize the FPU setting, vector table location and External memory configuration.

Variables

- const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}
- uint32_t SystemCoreClock = 16000000

6.15.1 Detailed Description

CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

This file provides two functions and one global variable to be called from user application:

- SystemInit(): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f4xx.s" file.
- SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
- SystemCoreClockUpdate(): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.

Attention

Index