

Gyro SPI with STM32F427VIT6 - ARTPART Assessment

1.0

Generated by Doxygen 1.9.8

1 Topic Index	1
1.1 Topics	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Topic Documentation	7
4.1 I3G4250D	7
4.1.1 Detailed Description	13
4.1.2 Macro Definition Documentation	14
4.1.2.1 __weak	14
4.1.2.2 I3G4250D_CTRL_REG1	14
4.1.2.3 I3G4250D_CTRL_REG2	14
4.1.2.4 I3G4250D_CTRL_REG3	14
4.1.2.5 I3G4250D_CTRL_REG4	14
4.1.2.6 I3G4250D_CTRL_REG5	14
4.1.2.7 I3G4250D_FIFO_CTRL_REG	14
4.1.2.8 I3G4250D_FIFO_SRC_REG	14
4.1.2.9 I3G4250D_INT1_CFG	14
4.1.2.10 I3G4250D_INT1_DURATION	14
4.1.2.11 I3G4250D_INT1_SRC	15
4.1.2.12 I3G4250D_INT1_TSH_XH	15
4.1.2.13 I3G4250D_INT1_TSH_XL	15
4.1.2.14 I3G4250D_INT1_TSH_YH	15
4.1.2.15 I3G4250D_INT1_TSH_YL	15
4.1.2.16 I3G4250D_INT1_TSH_ZH	15
4.1.2.17 I3G4250D_INT1_TSH_ZL	15
4.1.2.18 I3G4250D_OUT_TEMP	15
4.1.2.19 I3G4250D_OUT_X_H	15
4.1.2.20 I3G4250D_OUT_X_L	15
4.1.2.21 I3G4250D_OUT_Y_H	16
4.1.2.22 I3G4250D_OUT_Y_L	16
4.1.2.23 I3G4250D_OUT_Z_H	16
4.1.2.24 I3G4250D_OUT_Z_L	16
4.1.2.25 I3G4250D_REFERENCE	16
4.1.2.26 I3G4250D_STATUS_REG	16
4.1.2.27 I3G4250D_WHO_AM_I	16
4.1.3 Enumeration Type Documentation	16
4.1.3.1 i3g4250d_and_or_t	16
4.1.3.2 i3g4250d_ble_t	17

4.1.3.3 i3g4250d_bw_t	17
4.1.3.4 i3g4250d_dr_t	17
4.1.3.5 i3g4250d_fifo_mode_t	17
4.1.3.6 i3g4250d_fs_t	18
4.1.3.7 i3g4250d_h_lactive_t	18
4.1.3.8 i3g4250d_hpcf_t	18
4.1.3.9 i3g4250d_hpm_t	19
4.1.3.10 i3g4250d_int1_sel_t	19
4.1.3.11 i3g4250d_lir_t	19
4.1.3.12 i3g4250d_out_sel_t	19
4.1.3.13 i3g4250d_pp_od_t	19
4.1.3.14 i3g4250d_sim_t	20
4.1.3.15 i3g4250d_st_t	20
4.1.4 Function Documentation	20
4.1.4.1 i3g4250d_angular_rate_raw_get()	20
4.1.4.2 i3g4250d_axis_x_data_get()	21
4.1.4.3 i3g4250d_axis_x_data_set()	21
4.1.4.4 i3g4250d_axis_y_data_get()	21
4.1.4.5 i3g4250d_axis_y_data_set()	21
4.1.4.6 i3g4250d_axis_z_data_get()	22
4.1.4.7 i3g4250d_axis_z_data_set()	22
4.1.4.8 i3g4250d_boot_get()	22
4.1.4.9 i3g4250d_boot_set()	22
4.1.4.10 i3g4250d_data_format_get()	23
4.1.4.11 i3g4250d_data_format_set()	24
4.1.4.12 i3g4250d_data_rate_get()	24
4.1.4.13 i3g4250d_data_rate_set()	25
4.1.4.14 i3g4250d_device_id_get()	26
4.1.4.15 i3g4250d_fifo_data_level_get()	27
4.1.4.16 i3g4250d_fifo_empty_flag_get()	28
4.1.4.17 i3g4250d_fifo_enable_get()	28
4.1.4.18 i3g4250d_fifo_enable_set()	29
4.1.4.19 i3g4250d_fifo_mode_get()	30
4.1.4.20 i3g4250d_fifo_mode_set()	30
4.1.4.21 i3g4250d_fifo_ovr_flag_get()	32
4.1.4.22 i3g4250d_fifo_watermark_get()	33
4.1.4.23 i3g4250d_fifo_watermark_set()	33
4.1.4.24 i3g4250d_fifo_wtm_flag_get()	34
4.1.4.25 i3g4250d_filter_path_get()	35
4.1.4.26 i3g4250d_filter_path_internal_get()	35
4.1.4.27 i3g4250d_filter_path_internal_set()	36
4.1.4.28 i3g4250d_filter_path_set()	37

4.1.4.29 i3g4250d_flag_data_ready_get()	38
4.1.4.30 i3g4250d_from_fs245dps_to_mdps()	39
4.1.4.31 i3g4250d_from_lsb_to_celsius()	39
4.1.4.32 i3g4250d_full_scale_get()	39
4.1.4.33 i3g4250d_full_scale_set()	40
4.1.4.34 i3g4250d_hp_bandwidth_get()	41
4.1.4.35 i3g4250d_hp_bandwidth_set()	41
4.1.4.36 i3g4250d_hp_mode_get()	42
4.1.4.37 i3g4250d_hp_mode_set()	43
4.1.4.38 i3g4250d_hp_reference_value_get()	43
4.1.4.39 i3g4250d_hp_reference_value_set()	45
4.1.4.40 i3g4250d_int_notification_get()	46
4.1.4.41 i3g4250d_int_notification_set()	46
4.1.4.42 i3g4250d_int_on_threshold_conf_get()	48
4.1.4.43 i3g4250d_int_on_threshold_conf_set()	49
4.1.4.44 i3g4250d_int_on_threshold_dur_get()	49
4.1.4.45 i3g4250d_int_on_threshold_dur_set()	50
4.1.4.46 i3g4250d_int_on_threshold_mode_get()	51
4.1.4.47 i3g4250d_int_on_threshold_mode_set()	51
4.1.4.48 i3g4250d_int_on_threshold_src_get()	52
4.1.4.49 i3g4250d_int_x_threshold_get()	53
4.1.4.50 i3g4250d_int_x_threshold_set()	53
4.1.4.51 i3g4250d_int_y_threshold_get()	54
4.1.4.52 i3g4250d_int_y_threshold_set()	55
4.1.4.53 i3g4250d_int_z_threshold_get()	55
4.1.4.54 i3g4250d_int_z_threshold_set()	56
4.1.4.55 i3g4250d_lp_bandwidth_get()	57
4.1.4.56 i3g4250d_lp_bandwidth_set()	57
4.1.4.57 i3g4250d_pin_int1_route_get()	58
4.1.4.58 i3g4250d_pin_int1_route_set()	59
4.1.4.59 i3g4250d_pin_int2_route_get()	59
4.1.4.60 i3g4250d_pin_int2_route_set()	60
4.1.4.61 i3g4250d_pin_mode_get()	61
4.1.4.62 i3g4250d_pin_mode_set()	61
4.1.4.63 i3g4250d_pin_polarity_get()	63
4.1.4.64 i3g4250d_pin_polarity_set()	64
4.1.4.65 i3g4250d_read_reg()	64
4.1.4.66 i3g4250d_self_test_get()	65
4.1.4.67 i3g4250d_self_test_set()	66
4.1.4.68 i3g4250d_spi_mode_get()	67
4.1.4.69 i3g4250d_spi_mode_set()	67
4.1.4.70 i3g4250d_status_reg_get()	68

4.1.4.71 i3g4250d_temperature_raw_get()	68
4.1.4.72 i3g4250d_write_reg()	69
4.1.5 I3G4250D_Interfaces_Functions	71
4.1.5.1 Detailed Description	71
4.1.5.2 Function Documentation	71
4.1.6 I3G4250D_Sensitivity	73
4.1.6.1 Detailed Description	74
4.1.6.2 Function Documentation	74
4.1.7 I3G4250D_data_generation	75
4.1.7.1 Detailed Description	75
4.1.7.2 Function Documentation	75
4.1.8 I3G4250D_Dataoutput	81
4.1.8.1 Detailed Description	81
4.1.8.2 Function Documentation	82
4.1.9 I3G4250D_common	83
4.1.9.1 Detailed Description	84
4.1.9.2 Function Documentation	84
4.1.10 I3G4250D_filters	90
4.1.10.1 Detailed Description	91
4.1.10.2 Function Documentation	91
4.1.11 I3G4250D_serial_interface	101
4.1.11.1 Detailed Description	101
4.1.11.2 Function Documentation	101
4.1.12 I3G4250D_interrupt_pins	103
4.1.12.1 Detailed Description	104
4.1.12.2 Function Documentation	104
4.1.13 interrupt_on_threshold	111
4.1.13.1 Detailed Description	112
4.1.13.2 Function Documentation	112
4.1.14 I3G4250D_fifo	121
4.1.14.1 Detailed Description	122
4.1.14.2 Function Documentation	122
4.1.15 definitions	128
4.1.15.1 Detailed Description	128
4.1.15.2 Macro Definition Documentation	128
4.1.16 sensors common types	129
4.1.16.1 Detailed Description	130
4.1.16.2 Macro Definition Documentation	130
4.1.16.3 address-data structure definition	130
4.1.16.4 Interfaces_Functions	131
4.1.17 I3g4250d_Infos	132
4.1.17.1 Detailed Description	132

4.1.17.2 Macro Definition Documentation	132
4.1.18 LSM9DS1_Register_Union	132
4.1.18.1 Detailed Description	133
4.2 CMSIS	133
4.2.1 Detailed Description	134
4.2.2 Stm32f4xx_system	134
4.2.2.1 Detailed Description	134
4.2.2.2 STM32F4xx_System_Private_Includes	134
4.2.2.3 STM32F4xx_System_Private_TypesDefinitions	135
4.2.2.4 STM32F4xx_System_Private_Defines	135
4.2.2.5 STM32F4xx_System_Private_Macros	136
4.2.2.6 STM32F4xx_System_Private_Variables	136
4.2.2.7 STM32F4xx_System_Private_FunctionPrototypes	137
4.2.2.8 STM32F4xx_System_Private_Functions	137
5 Data Structure Documentation	141
5.1 bitwise_t Struct Reference	141
5.1.1 Field Documentation	141
5.1.1.1 bit0	141
5.1.1.2 bit1	141
5.1.1.3 bit2	141
5.1.1.4 bit3	142
5.1.1.5 bit4	142
5.1.1.6 bit5	142
5.1.1.7 bit6	142
5.1.1.8 bit7	142
5.2 i3g4250d_ctrl_reg1_t Struct Reference	142
5.2.1 Field Documentation	142
5.2.1.1 bw	142
5.2.1.2 dr	143
5.2.1.3 pd	143
5.3 i3g4250d_ctrl_reg2_t Struct Reference	143
5.3.1 Field Documentation	143
5.3.1.1 hpcf	143
5.3.1.2 hpm	143
5.3.1.3 not_used_01	143
5.4 i3g4250d_ctrl_reg3_t Struct Reference	144
5.4.1 Field Documentation	144
5.4.1.1 h_lactive	144
5.4.1.2 i1_boot	144
5.4.1.3 i1_int1	144
5.4.1.4 i2_drdy	144

5.4.1.5 i2_empty	144
5.4.1.6 i2_orun	145
5.4.1.7 i2_wtm	145
5.4.1.8 pp_od	145
5.5 i3g4250d_ctrl_reg4_t Struct Reference	145
5.5.1 Field Documentation	145
5.5.1.1 ble	145
5.5.1.2 fs	146
5.5.1.3 not_used_01	146
5.5.1.4 not_used_02	146
5.5.1.5 sim	146
5.5.1.6 st	146
5.6 i3g4250d_ctrl_reg5_t Struct Reference	146
5.6.1 Field Documentation	147
5.6.1.1 boot	147
5.6.1.2 fifo_en	147
5.6.1.3 hpen	147
5.6.1.4 int1_sel	147
5.6.1.5 not_used_01	147
5.6.1.6 out_sel	147
5.7 i3g4250d_fifo_ctrl_reg_t Struct Reference	147
5.7.1 Field Documentation	148
5.7.1.1 fm	148
5.7.1.2 wtm	148
5.8 i3g4250d_fifo_src_reg_t Struct Reference	148
5.8.1 Field Documentation	148
5.8.1.1 empty	148
5.8.1.2 fss	148
5.8.1.3 ovrn	149
5.8.1.4 wtm	149
5.9 i3g4250d_int1_cfg_t Struct Reference	149
5.9.1 Field Documentation	149
5.9.1.1 and_or	149
5.9.1.2 lir	149
5.9.1.3 xhie	150
5.9.1.4 xlie	150
5.9.1.5 yhie	150
5.9.1.6 ylie	150
5.9.1.7 zhie	150
5.9.1.8 zlie	150
5.10 i3g4250d_int1_duration_t Struct Reference	150
5.10.1 Field Documentation	150

5.10.1.1 d	150
5.10.1.2 wait	151
5.11 i3g4250d_int1_route_t Struct Reference	151
5.11.1 Field Documentation	151
5.11.1.1 i1_boot	151
5.11.1.2 i1_int1	151
5.12 i3g4250d_int1_src_t Struct Reference	151
5.12.1 Field Documentation	152
5.12.1.1 ia	152
5.12.1.2 not_used_01	152
5.12.1.3 xh	152
5.12.1.4 xl	152
5.12.1.5 yh	152
5.12.1.6 yl	152
5.12.1.7 zh	152
5.12.1.8 zl	153
5.13 i3g4250d_int1_tsh_xh_t Struct Reference	153
5.13.1 Field Documentation	153
5.13.1.1 not_used_01	153
5.13.1.2 thsx	153
5.14 i3g4250d_int1_tsh_xl_t Struct Reference	153
5.14.1 Field Documentation	154
5.14.1.1 thsx	154
5.15 i3g4250d_int1_tsh_yh_t Struct Reference	154
5.15.1 Field Documentation	154
5.15.1.1 not_used_01	154
5.15.1.2 thsy	154
5.16 i3g4250d_int1_tsh_yl_t Struct Reference	154
5.16.1 Field Documentation	155
5.16.1.1 thsy	155
5.17 i3g4250d_int1_tsh_zh_t Struct Reference	155
5.17.1 Field Documentation	155
5.17.1.1 not_used_01	155
5.17.1.2 thsz	155
5.18 i3g4250d_int1_tsh_zl_t Struct Reference	155
5.18.1 Field Documentation	156
5.18.1.1 thsz	156
5.19 i3g4250d_int2_route_t Struct Reference	156
5.19.1 Field Documentation	156
5.19.1.1 i2_drdy	156
5.19.1.2 i2_empty	156
5.19.1.3 i2_orun	156

5.19.1.4 i2_wtm	157
5.20 i3g4250d_reference_t Struct Reference	157
5.20.1 Field Documentation	157
5.20.1.1 ref	157
5.21 i3g4250d_reg_t Union Reference	157
5.21.1 Field Documentation	159
5.21.1.1 bitwise	159
5.21.1.2 byte	159
5.21.1.3 ctrl_reg1	159
5.21.1.4 ctrl_reg2	159
5.21.1.5 ctrl_reg3	159
5.21.1.6 ctrl_reg4	159
5.21.1.7 ctrl_reg5	159
5.21.1.8 fifo_ctrl_reg	160
5.21.1.9 fifo_src_reg	160
5.21.1.10 int1_cfg	160
5.21.1.11 int1_duration	160
5.21.1.12 int1_src	160
5.21.1.13 int1_tsh_xh	160
5.21.1.14 int1_tsh_xl	160
5.21.1.15 int1_tsh_yh	160
5.21.1.16 int1_tsh_yl	160
5.21.1.17 int1_tsh_zh	160
5.21.1.18 int1_tsh_zl	161
5.21.1.19 reference	161
5.21.1.20 status_reg	161
5.22 i3g4250d_status_reg_t Struct Reference	161
5.22.1 Field Documentation	161
5.22.1.1 _xor	161
5.22.1.2 xda	161
5.22.1.3 yda	161
5.22.1.4 yor	162
5.22.1.5 zda	162
5.22.1.6 zor	162
5.22.1.7 zyxda	162
5.22.1.8 zyxor	162
5.23 stmdev_ctx_t Struct Reference	162
5.23.1 Field Documentation	162
5.23.1.1 handle	162
5.23.1.2 mdelay	163
5.23.1.3 read_reg	163
5.23.1.4 write_reg	163

5.24 ucf_line_t Struct Reference	163
5.24.1 Field Documentation	163
5.24.1.1 address	163
5.24.1.2 data	163
6 File Documentation	165
6.1 i3g4250d_reg.h File Reference	165
6.1.1 Detailed Description	170
6.2 i3g4250d_reg.h	171
6.3 main.h File Reference	179
6.3.1 Detailed Description	179
6.3.2 Macro Definition Documentation	180
6.3.2.1 GYRO_CS_OUT_GPIO_Port	180
6.3.2.2 GYRO_CS_OUT_Pin	180
6.3.3 Function Documentation	180
6.3.3.1 Error_Handler()	180
6.4 main.h	180
6.5 stm32f4xx_hal_conf.h File Reference	181
6.5.1 Macro Definition Documentation	183
6.5.1.1 assert_param	183
6.5.1.2 DATA_CACHE_ENABLE	184
6.5.1.3 DP83848_PHY_ADDRESS	184
6.5.1.4 ETH_RX_BUF_SIZE	184
6.5.1.5 ETH_RXBUFNB	184
6.5.1.6 ETH_TX_BUF_SIZE	184
6.5.1.7 ETH_TXBUFNB	184
6.5.1.8 EXTERNAL_CLOCK_VALUE	184
6.5.1.9 HAL_CORTEX_MODULE_ENABLED	184
6.5.1.10 HAL_DMA_MODULE_ENABLED	184
6.5.1.11 HAL_EXTI_MODULE_ENABLED	185
6.5.1.12 HAL_FLASH_MODULE_ENABLED	185
6.5.1.13 HAL_GPIO_MODULE_ENABLED	185
6.5.1.14 HAL_MODULE_ENABLED	185
6.5.1.15 HAL_PWR_MODULE_ENABLED	185
6.5.1.16 HAL_RCC_MODULE_ENABLED	185
6.5.1.17 HAL_SPI_MODULE_ENABLED	185
6.5.1.18 HAL_UART_MODULE_ENABLED	185
6.5.1.19 HSE_STARTUP_TIMEOUT	185
6.5.1.20 HSE_VALUE	186
6.5.1.21 HSI_VALUE	186
6.5.1.22 INSTRUCTION_CACHE_ENABLE	186
6.5.1.23 LSE_STARTUP_TIMEOUT	186

6.5.1.24 LSE_VALUE	186
6.5.1.25 LSI_VALUE	186
6.5.1.26 MAC_ADDR0	187
6.5.1.27 MAC_ADDR1	187
6.5.1.28 MAC_ADDR2	187
6.5.1.29 MAC_ADDR3	187
6.5.1.30 MAC_ADDR4	187
6.5.1.31 MAC_ADDR5	187
6.5.1.32 PHY_AUTONEGO_COMPLETE	187
6.5.1.33 PHY_AUTONEGOTIATION	187
6.5.1.34 PHY_BCR	187
6.5.1.35 PHY_BSR	188
6.5.1.36 PHY_CONFIG_DELAY	188
6.5.1.37 PHY_DUPLEX_STATUS	188
6.5.1.38 PHY_FULLDUPLEX_100M	188
6.5.1.39 PHY_FULLDUPLEX_10M	188
6.5.1.40 PHY_HALFDUPLEX_100M	188
6.5.1.41 PHY_HALFDUPLEX_10M	188
6.5.1.42 PHY_ISOLATE	189
6.5.1.43 PHY_JABBER_DETECTION	189
6.5.1.44 PHY_LINKED_STATUS	189
6.5.1.45 PHY_LOOPBACK	189
6.5.1.46 PHY_POWERDOWN	189
6.5.1.47 PHY_READ_TO	189
6.5.1.48 PHY_RESET	189
6.5.1.49 PHY_RESET_DELAY	190
6.5.1.50 PHY_RESTART_AUTONEGOTIATION	190
6.5.1.51 PHY_SPEED_STATUS	190
6.5.1.52 PHY_SR	190
6.5.1.53 PHY_WRITE_TO	190
6.5.1.54 PREFETCH_ENABLE	190
6.5.1.55 TICK_INT_PRIORITY	190
6.5.1.56 USE_HAL_ADC_REGISTER_CALLBACKS	190
6.5.1.57 USE_HAL_CAN_REGISTER_CALLBACKS	191
6.5.1.58 USE_HAL_CEC_REGISTER_CALLBACKS	191
6.5.1.59 USE_HAL_Cryp_REGISTER_CALLBACKS	191
6.5.1.60 USE_HAL_DAC_REGISTER_CALLBACKS	191
6.5.1.61 USE_HAL_DCMI_REGISTER_CALLBACKS	191
6.5.1.62 USE_HAL_DFSDM_REGISTER_CALLBACKS	191
6.5.1.63 USE_HAL_DMA2D_REGISTER_CALLBACKS	191
6.5.1.64 USE_HAL_DSI_REGISTER_CALLBACKS	191
6.5.1.65 USE_HAL_ETH_REGISTER_CALLBACKS	191

6.5.1.66 USE_HAL_FMPI2C_REGISTER_CALLBACKS	191
6.5.1.67 USE_HAL_FMPSPMBUS_REGISTER_CALLBACKS	192
6.5.1.68 USE_HAL_HASH_REGISTER_CALLBACKS	192
6.5.1.69 USE_HAL_HCD_REGISTER_CALLBACKS	192
6.5.1.70 USE_HAL_I2C_REGISTER_CALLBACKS	192
6.5.1.71 USE_HAL_I2S_REGISTER_CALLBACKS	192
6.5.1.72 USE_HAL_IRDA_REGISTER_CALLBACKS	192
6.5.1.73 USE_HAL_LPTIM_REGISTER_CALLBACKS	192
6.5.1.74 USE_HAL_LTDC_REGISTER_CALLBACKS	192
6.5.1.75 USE_HAL_MMC_REGISTER_CALLBACKS	192
6.5.1.76 USE_HAL_NAND_REGISTER_CALLBACKS	192
6.5.1.77 USE_HAL_NOR_REGISTER_CALLBACKS	193
6.5.1.78 USE_HAL_PCCARD_REGISTER_CALLBACKS	193
6.5.1.79 USE_HAL_PCD_REGISTER_CALLBACKS	193
6.5.1.80 USE_HAL_QSPI_REGISTER_CALLBACKS	193
6.5.1.81 USE_HAL_RNG_REGISTER_CALLBACKS	193
6.5.1.82 USE_HAL_RTC_REGISTER_CALLBACKS	193
6.5.1.83 USE_HAL_SAI_REGISTER_CALLBACKS	193
6.5.1.84 USE_HAL_SD_REGISTER_CALLBACKS	193
6.5.1.85 USE_HAL_SDRAM_REGISTER_CALLBACKS	193
6.5.1.86 USE_HAL_SMARTCARD_REGISTER_CALLBACKS	193
6.5.1.87 USE_HAL_SMBUS_REGISTER_CALLBACKS	194
6.5.1.88 USE_HAL_SPDIFRX_REGISTER_CALLBACKS	194
6.5.1.89 USE_HAL_SPI_REGISTER_CALLBACKS	194
6.5.1.90 USE_HAL_SRAM_REGISTER_CALLBACKS	194
6.5.1.91 USE_HAL_TIM_REGISTER_CALLBACKS	194
6.5.1.92 USE_HAL_UART_REGISTER_CALLBACKS	194
6.5.1.93 USE_HAL_USART_REGISTER_CALLBACKS	194
6.5.1.94 USE_HAL_WWDG_REGISTER_CALLBACKS	194
6.5.1.95 USE_RTOS	194
6.5.1.96 USE_SPI_CRC	194
6.5.1.97 VDD_VALUE	195
6.6 stm32f4xx_hal_conf.h	195
6.7 stm32f4xx_it.h File Reference	200
6.7.1 Detailed Description	201
6.7.2 Function Documentation	201
6.7.2.1 BusFault_Handler()	201
6.7.2.2 DebugMon_Handler()	201
6.7.2.3 HardFault_Handler()	201
6.7.2.4 MemManage_Handler()	201
6.7.2.5 NMI_Handler()	201
6.7.2.6 PendSV_Handler()	202

6.7.2.7 SVC_Handler()	202
6.7.2.8 SysTick_Handler()	202
6.7.2.9 UsageFault_Handler()	202
6.8 stm32f4xx_it.h	202
6.9 i3g4250d_reg.c File Reference	203
6.9.1 Detailed Description	206
6.10 main.c File Reference	207
6.10.1 Detailed Description	207
6.10.2 Macro Definition Documentation	208
6.10.2.1 BOOT_TIME	208
6.10.3 Function Documentation	208
6.10.3.1 main()	208
6.10.3.2 SystemClock_Config()	210
6.10.4 Variable Documentation	211
6.10.4.1 hspi1	211
6.10.4.2 huart2	211
6.11 stm32f4xx_hal_msp.c File Reference	211
6.11.1 Detailed Description	212
6.11.2 Function Documentation	212
6.11.2.1 HAL_MspInit()	212
6.11.2.2 HAL_SPI_MspDeInit()	212
6.11.2.3 HAL_SPI_MspInit()	213
6.11.2.4 HAL_UART_MspDeInit()	213
6.11.2.5 HAL_UART_MspInit()	213
6.12 stm32f4xx_it.c File Reference	214
6.12.1 Detailed Description	215
6.12.2 Function Documentation	215
6.12.2.1 BusFault_Handler()	215
6.12.2.2 DebugMon_Handler()	215
6.12.2.3 HardFault_Handler()	215
6.12.2.4 MemManage_Handler()	215
6.12.2.5 NMI_Handler()	215
6.12.2.6 PendSV_Handler()	216
6.12.2.7 SVC_Handler()	216
6.12.2.8 SysTick_Handler()	216
6.12.2.9 UsageFault_Handler()	216
6.13 syscalls.c File Reference	216
6.13.1 Detailed Description	217
6.13.2 Function Documentation	218
6.13.2.1 __attribute__()	218
6.13.2.2 __io_getchar()	218
6.13.2.3 __io_putchar()	218

6.13.2.4 _close()	218
6.13.2.5 _execve()	219
6.13.2.6 _exit()	219
6.13.2.7 _fork()	219
6.13.2.8 _fstat()	219
6.13.2.9 _getpid()	219
6.13.2.10 _isatty()	219
6.13.2.11 _kill()	220
6.13.2.12 _link()	220
6.13.2.13 _lseek()	220
6.13.2.14 _open()	220
6.13.2.15 _stat()	220
6.13.2.16 _times()	220
6.13.2.17 _unlink()	221
6.13.2.18 _wait()	221
6.13.2.19 initialise_monitor_handles()	221
6.13.3 Variable Documentation	221
6.13.3.1 environ	221
6.14 system.c File Reference	221
6.14.1 Detailed Description	222
6.14.2 Function Documentation	222
6.14.2.1 _sbrk()	222
6.15 system_stm32f4xx.c File Reference	223
6.15.1 Detailed Description	224
Index	225

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

I3G4250D	7
I3G4250D_Interfaces_Functions	71
I3G4250D_Sensitivity	73
I3G4250D_data_generation	75
I3G4250D_Dataoutput	81
I3G4250D_common	83
I3G4250D_filters	90
I3G4250D_serial_interface	101
I3G4250D_interrupt_pins	103
interrupt_on_threshold	111
I3G4250D_fifo	121
definitions	128
sensors common types	129
address-data structure definition	130
Interfaces_Functions	131
I3g4250d_Infos	132
LSM9DS1_Register_Union	132
CMSIS	133
Stm32f4xx_system	134
STM32F4xx_System_Private_Includes	134
STM32F4xx_System_Private_TypesDefinitions	135
STM32F4xx_System_Private_Defines	135
STM32F4xx_System_Private_Macros	136
STM32F4xx_System_Private_Variables	136
STM32F4xx_System_Private_FunctionPrototypes	137
STM32F4xx_System_Private_Functions	137

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

bitwise_t	141
i3g4250d_ctrl_reg1_t	142
i3g4250d_ctrl_reg2_t	143
i3g4250d_ctrl_reg3_t	144
i3g4250d_ctrl_reg4_t	145
i3g4250d_ctrl_reg5_t	146
i3g4250d_fifo_ctrl_reg_t	147
i3g4250d_fifo_src_reg_t	148
i3g4250d_int1_cfg_t	149
i3g4250d_int1_duration_t	150
i3g4250d_int1_route_t	151
i3g4250d_int1_src_t	151
i3g4250d_int1_tsh_xh_t	153
i3g4250d_int1_tsh_xl_t	153
i3g4250d_int1_tsh_yh_t	154
i3g4250d_int1_tsh_yl_t	154
i3g4250d_int1_tsh_zh_t	155
i3g4250d_int1_tsh_zl_t	155
i3g4250d_int2_route_t	156
i3g4250d_reference_t	157
i3g4250d_reg_t	157
i3g4250d_status_reg_t	161
stmdev_ctx_t	162
ucf_line_t	163

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

i3g4250d_reg.h	This file contains all the functions prototypes for the i3g4250d_reg.c driver	165
main.h	: Header for main.c file. This file contains the common defines of the application	179
stm32f4xx_hal_conf.h	181
stm32f4xx_it.h	This file contains the headers of the interrupt handlers	200
i3g4250d_reg.c	I3G4250D driver file	203
main.c	: Main program body	207
stm32f4xx_hal_msp.c	This file provides code for the MSP Initialization and de-Initialization codes	211
stm32f4xx_it.c	Interrupt Service Routines	214
syscalls.c	STM32CubeIDE Minimal System calls file	216
systemem.c	STM32CubeIDE System Memory calls file	221
system_stm32f4xx.c	CMSIS Cortex-M4 Device Peripheral Access Layer System Source File	223

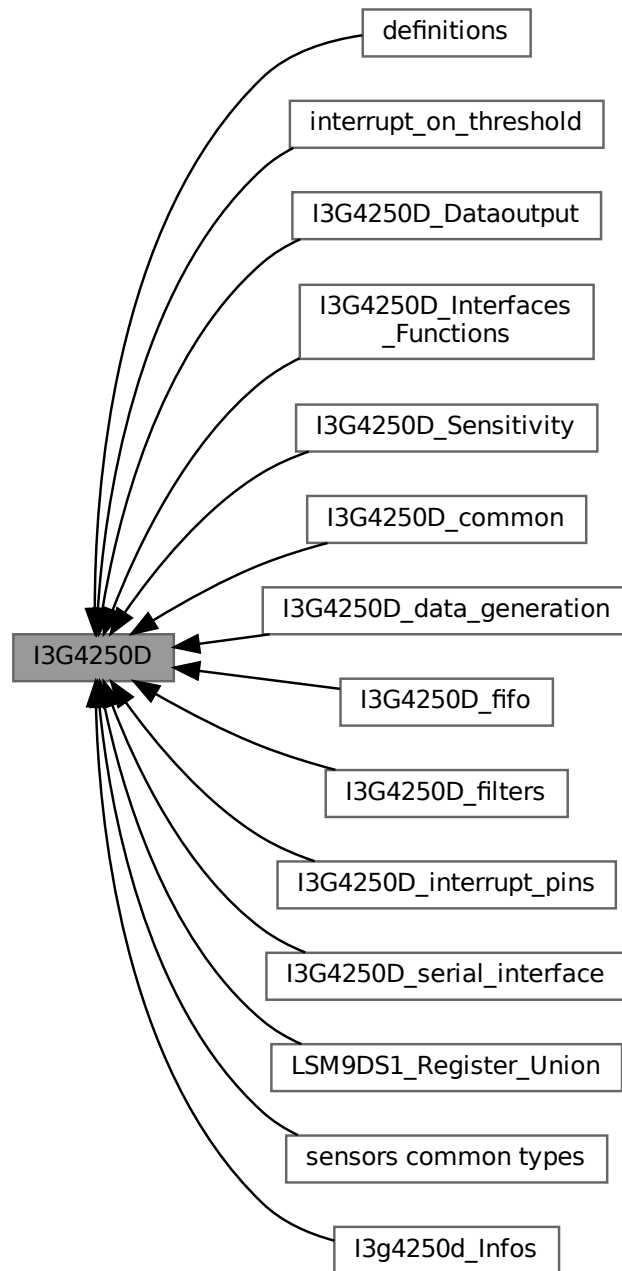
Chapter 4

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_Funcions](#)

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 am I.[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)
Overflow 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

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

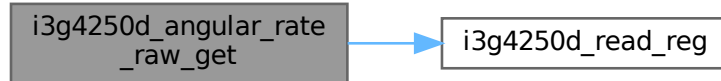
Return values

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

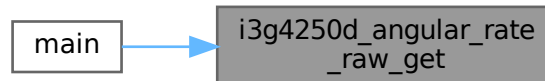
References [I3G4250D_OUT_X_L](#), and [i3g4250d_read_reg\(\)](#).

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

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

Return values

<i>interface</i>	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

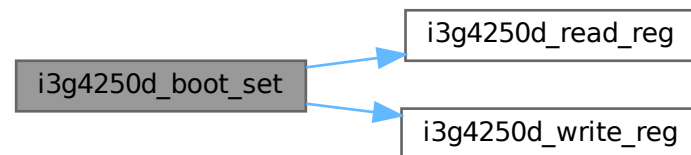
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of boot in reg CTRL_REG5.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

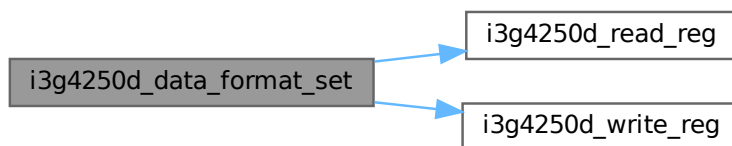
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "ble" in reg CTRL_REG4.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

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

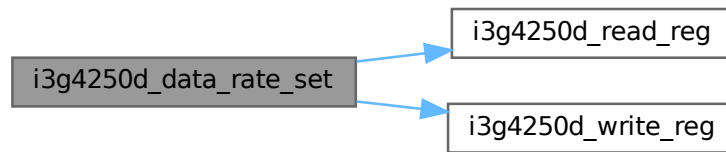
Return values

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

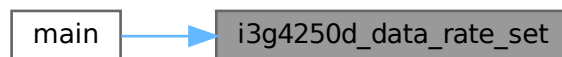
References [i3g4250d_ctrl_reg1_t::dr](#), [I3G4250D_CTRL_REG1](#), [i3g4250d_read_reg\(\)](#), [i3g4250d_write_reg\(\)](#), and [i3g4250d_ctrl_reg1_t::pd](#).

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 `aml.[get]`.

Parameters

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

Return values

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

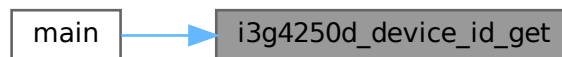
References [i3g4250d_read_reg\(\)](#), and [I3G4250D_WHO_AM_I](#).

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

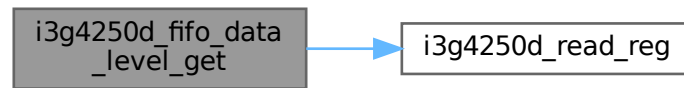
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of fss in reg FIFO_SRC_REG.(ptr)

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

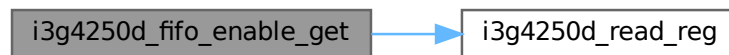
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of fifo_en in reg CTRL_REG5.(ptr)

Return values

<i>Interface</i>	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

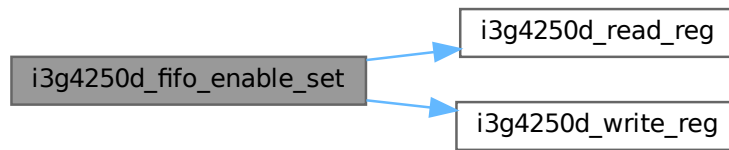
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of fifo_en in reg CTRL_REG5

Return values

<i>Interface</i>	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

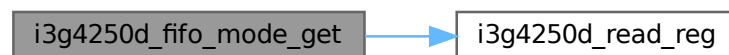
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of fm in reg FIFO_CTRL_REG.(ptr)

Return values

<i>Interface</i>	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

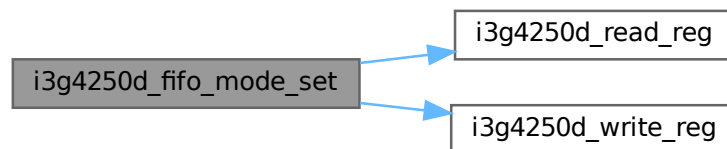
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of fm in reg FIFO_CTRL_REG

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

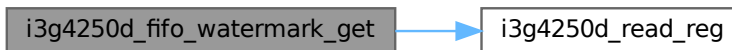
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of wtm in reg FIFO_CTRL_REG.(ptr)

Return values

<i>Interface</i>	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

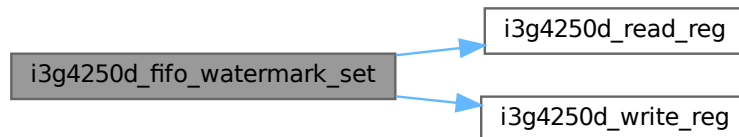
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of wtm in reg FIFO_CTRL_REG

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

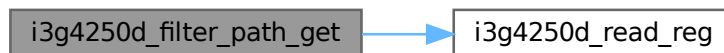
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of out_sel in reg CTRL_REG5.(ptr)

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

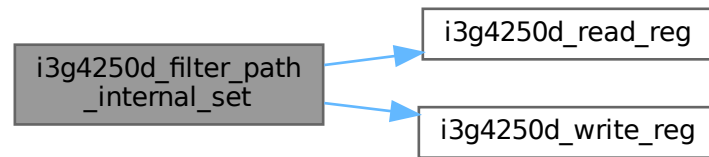
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of int1_sel in reg CTRL_REG5

Return values

<i>Interface</i>	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

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

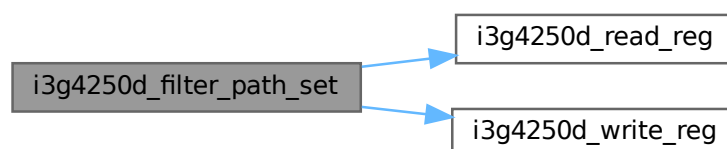
Return values

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

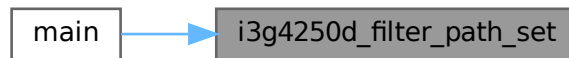
References [i3g4250d_ctrl_reg5_t::hpen](#), [I3G4250D_CTRL_REG5](#), [i3g4250d_read_reg\(\)](#), [i3g4250d_write_reg\(\)](#), and [i3g4250d_ctrl_reg5_t::out_sel](#).

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

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

Return values

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

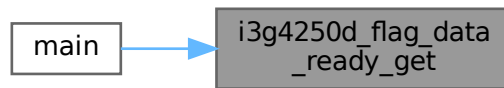
References [i3g4250d_read_reg\(\)](#), [I3G4250D_STATUS_REG](#), and [i3g4250d_status_reg_t::zyxda](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

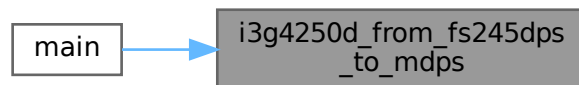


4.1.4.30 i3g4250d_from_fs245dps_to_mdps()

```
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

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

Return values

<i>Interface</i>	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

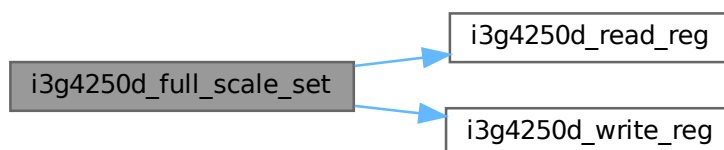
<i>ctx</i>	read / write interface definitions(ptr)
<i>val</i>	change the values of fs in reg CTRL_REG4

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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:



4.1.4.35 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

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

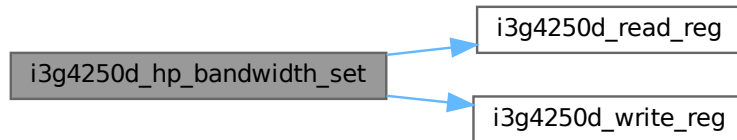
Return values

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

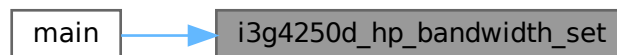
References [i3g4250d_ctrl_reg2_t::hpcf](#), [I3G4250D_CTRL_REG2](#), [i3g4250d_read_reg\(\)](#), and [i3g4250d_write_reg\(\)](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.4.36 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

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

Return values

<i>Interface</i>	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

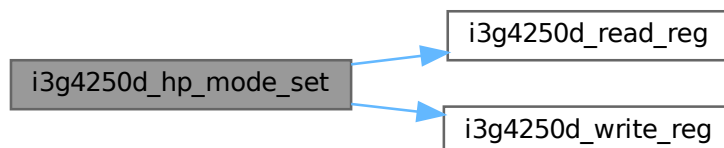
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "hpm" in reg CTRL_REG2.

Return values

<i>Interface</i>	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

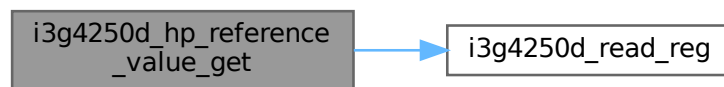
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of ref in reg REFERENCE.(ptr)

Return values

<i>Interface</i>	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

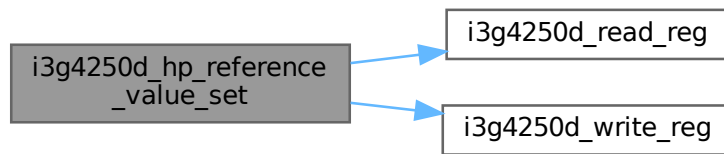
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of ref in reg REFERENCE

Return values

<i>Interface</i>	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

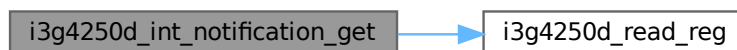
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of lir in reg INT1_CFG.(ptr)

Return values

<i>Interface</i>	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

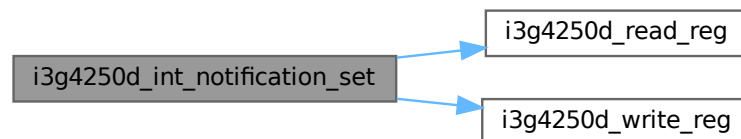
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of lir in reg INT1_CFG.

Return values

<i>Interface</i>	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

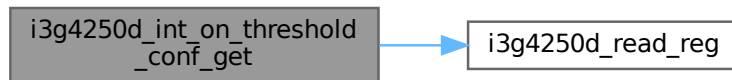
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Struct of registers from INT1_CFG to.(ptr)

Return values

<i>Interface</i>	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

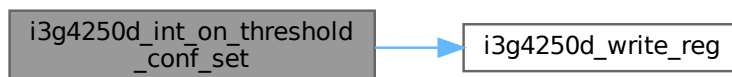
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Struct of registers INT1_CFG

Return values

<i>Interface</i>	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

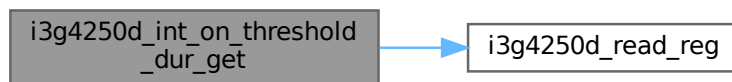
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of d in reg INT1_DURATION.(ptr)

Return values

<i>Interface</i>	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

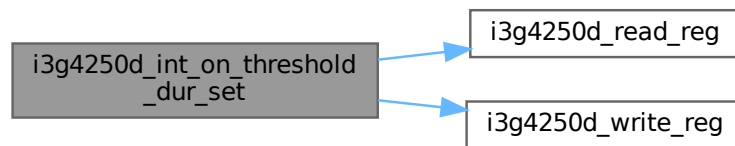
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of d in reg INT1_DURATION

Return values

<i>Interface</i>	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

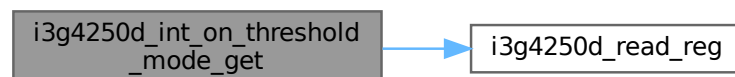
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of and_or in reg INT1_CFG.(ptr)

Return values

<i>Interface</i>	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

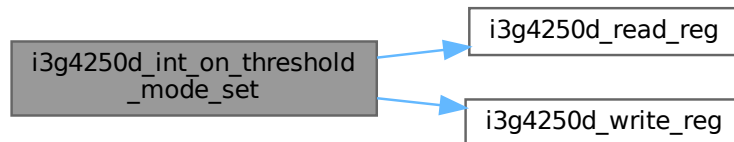
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of and_or in reg INT1_CFG

Return values

<i>Interface</i>	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

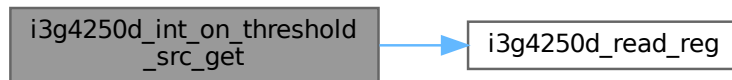
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Union of registers from INT1_SRC to.(ptr)

Return values

<i>Interface</i>	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

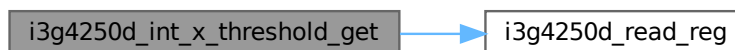
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsx in reg INT1_TSH_XH.(ptr)

Return values

<i>Interface</i>	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:



4.1.4.50 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

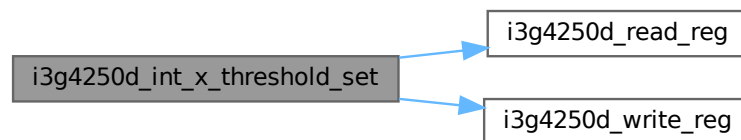
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsx in reg INT1_TSH_XH

Return values

<i>Interface</i>	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

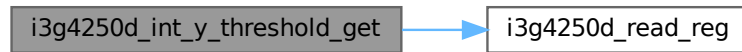
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsy in reg INT1_TSH_YH.(ptr)

Return values

<i>Interface</i>	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

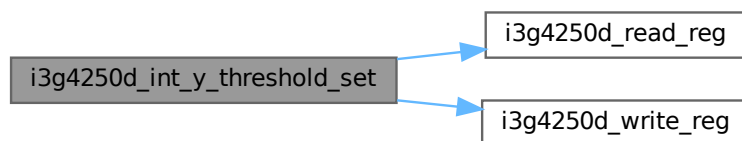
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsy in reg INT1_TSH_YH

Return values

<i>Interface</i>	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

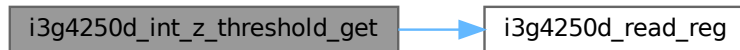
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsz in reg INT1_TSH_ZH.(ptr)

Return values

<i>Interface</i>	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:

**4.1.4.54 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

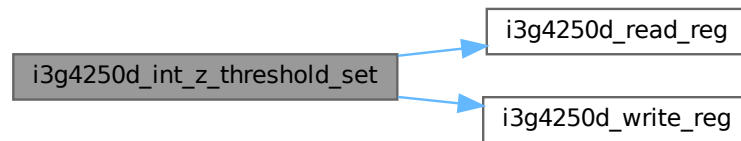
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsz in reg INT1_TSH_ZH.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

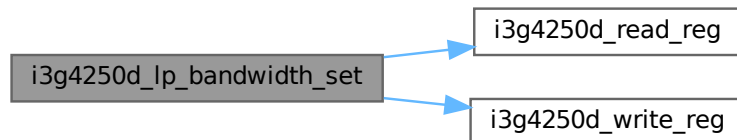
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "bw" in reg CTRL_REG1.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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:



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

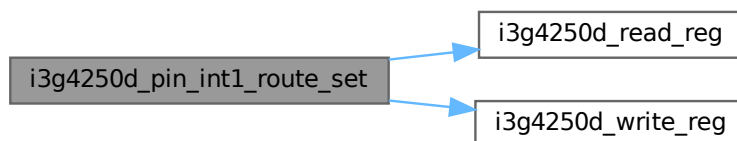
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Configure CTRL_REG3 int1 pad

Return values

<i>Interface</i>	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

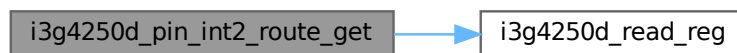
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Read CTRL_REG3 int2 pad.(ptr)

Return values

<i>Interface</i>	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:



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

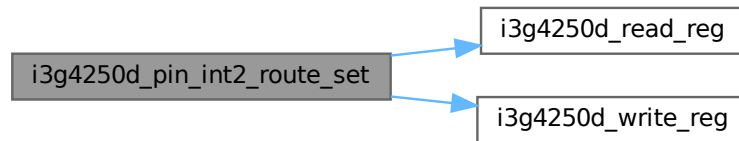
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Configure CTRL_REG3 int2 pad

Return values

<i>Interface</i>	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

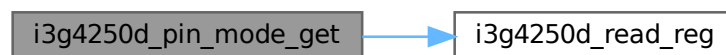
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of pp_od in reg CTRL_REG3.(ptr)

Return values

<i>Interface</i>	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

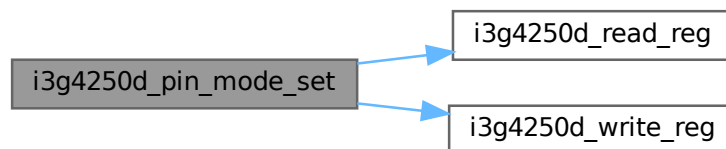
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of pp_od in reg CTRL_REG3

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

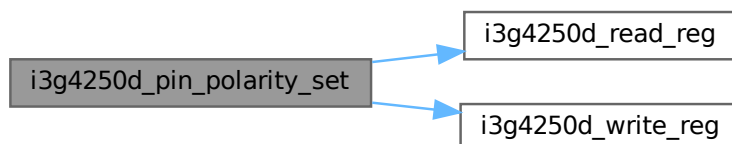
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of h_lactive in reg CTRL_REG3.

Return values

<i>Interface</i>	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

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

Return values

<i>interface</i>	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_get()`, `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

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

Return values

<i>Interface</i>	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

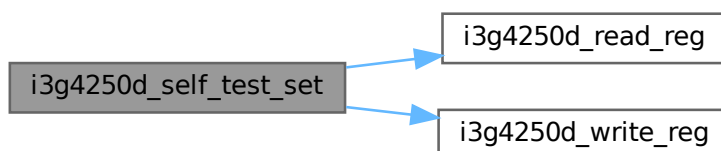
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	change the values of st in reg CTRL_REG4.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

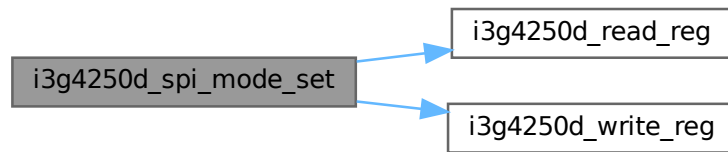
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of sim in reg CTRL_REG4

Return values

<i>Interface</i>	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

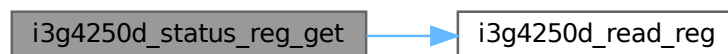
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	registers STATUS_REG

Return values

<i>Interface</i>	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

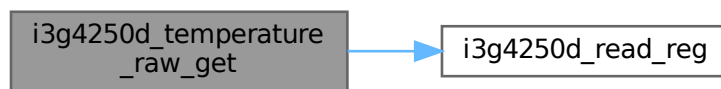
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>buff</i>	Buffer that stores the data read.(ptr)

Return values

<i>Interface</i>	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

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

Return values

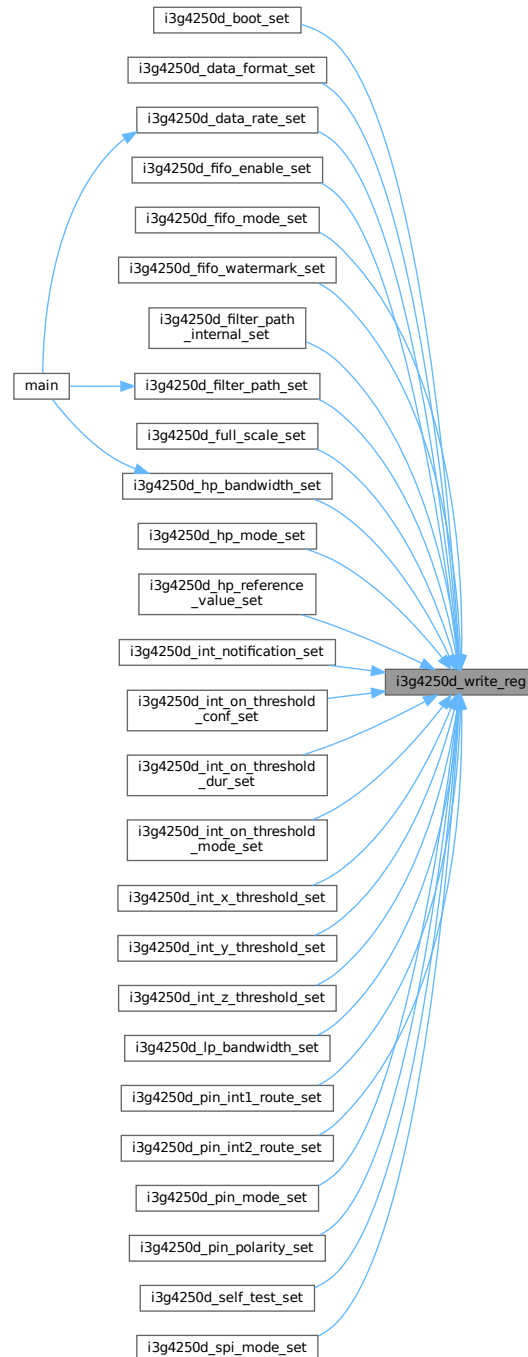
<i>interface</i>	status (MANDATORY: return 0 -> no Error)
------------------	--

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\(\)](#).

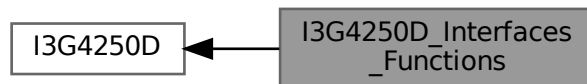
Here is the caller graph for this function:



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

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

Return values

<i>interface</i>	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_get\(\)](#), [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

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

Return values

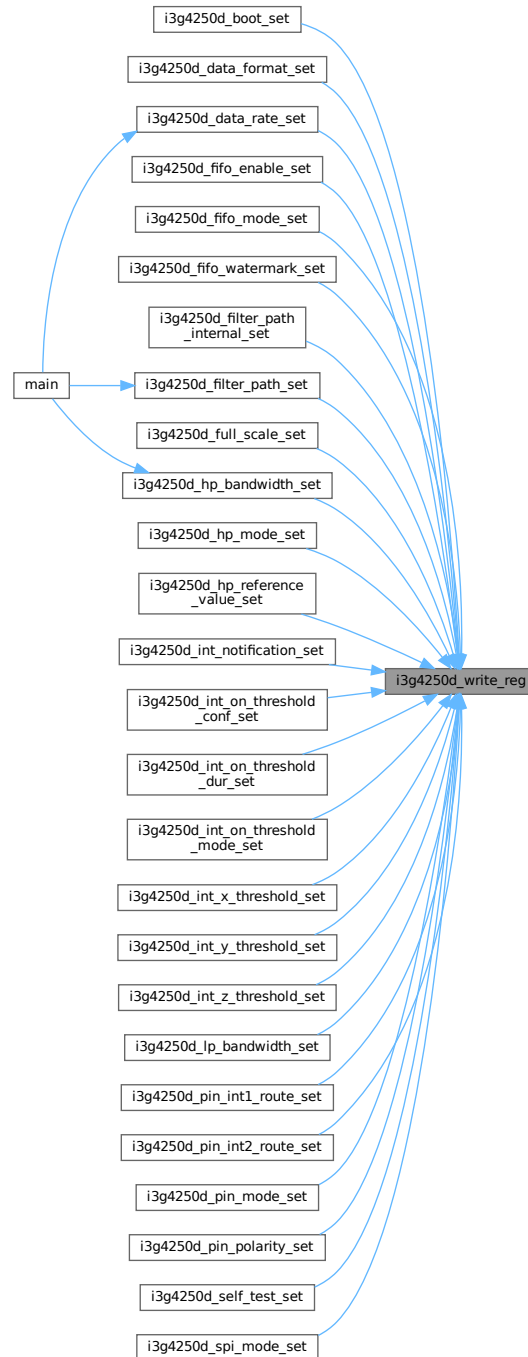
<i>interface</i>	status (MANDATORY: return 0 -> no Error)
------------------	--

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\(\)](#).

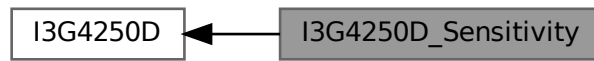
Here is the caller graph for this function:



4.1.6 I3G4250D_Sensitivity

These functions convert raw-data into engineering units.

Collaboration diagram for 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.

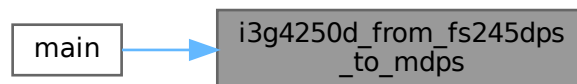
4.1.6.2 Function Documentation

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:



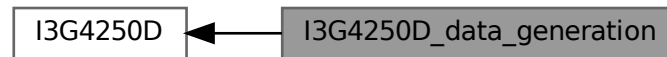
4.1.6.2.2 i3g4250d_from_lsb_to_celsius()

```
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

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

Return values

<i>Interface</i>	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

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

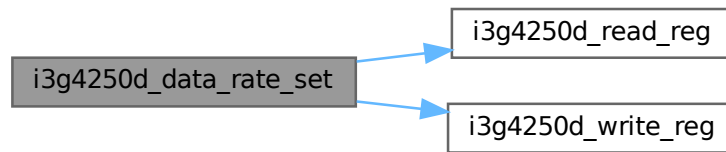
Return values

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

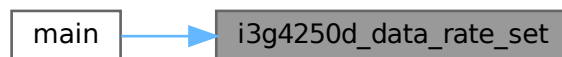
References [i3g4250d_ctrl_reg1_t::dr](#), [I3G4250D_CTRL_REG1](#), [i3g4250d_read_reg\(\)](#), [i3g4250d_write_reg\(\)](#), and [i3g4250d_ctrl_reg1_t::pd](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.7.2.3 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

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

Return values

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

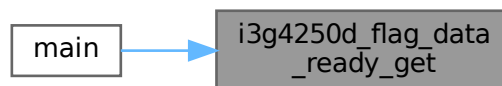
References [i3g4250d_read_reg\(\)](#), [I3G4250D_STATUS_REG](#), and [i3g4250d_status_reg_t::zyxda](#).

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

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

Return values

<i>Interface</i>	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

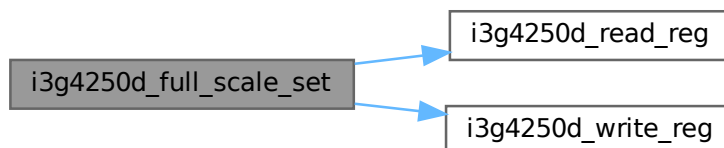
<i>ctx</i>	read / write interface definitions(ptr)
<i>val</i>	change the values of fs in reg CTRL_REG4

Return values

<i>Interface</i>	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

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

Return values

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

References [i3g4250d_read_reg\(\)](#), and [I3G4250D_STATUS_REG](#).

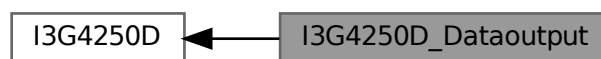
Here is the call graph for this function:



4.1.8 I3G4250D_Dataoutput

This section groups all the data output functions.

Collaboration diagram for I3G4250D_Dataoutput:



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

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

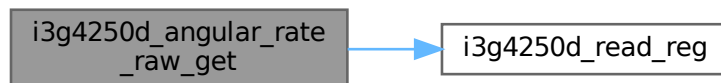
Return values

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

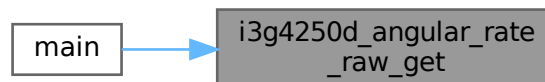
References [I3G4250D_OUT_X_L](#), and [i3g4250d_read_reg\(\)](#).

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

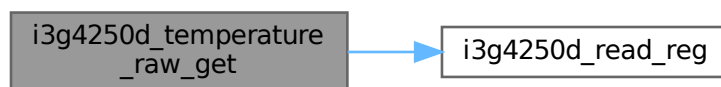
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>buff</i>	Buffer that stores the data read.(ptr)

Return values

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

References [I3G4250D_OUT_TEMP](#), and [i3g4250d_read_reg\(\)](#).

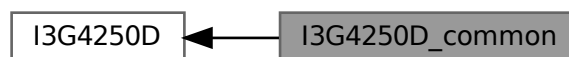
Here is the call graph for this function:



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 aml.[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

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

Return values

<i>interface</i>	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

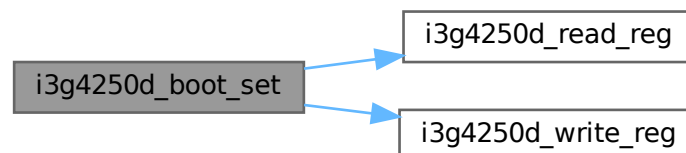
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of boot in reg CTRL_REG5.

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

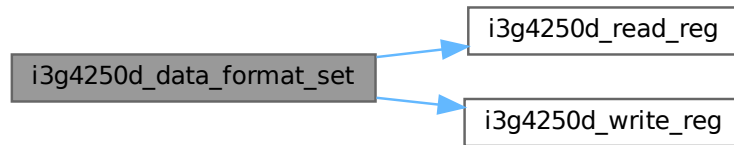
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "ble" in reg CTRL_REG4.

Return values

<i>Interface</i>	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 aml.[get].

Parameters

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

Return values

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

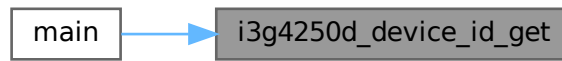
References [i3g4250d_read_reg\(\)](#), and [I3G4250D_WHO_AM_I](#).

Referenced by [main\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.9.2.6 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

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

Return values

<i>Interface</i>	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

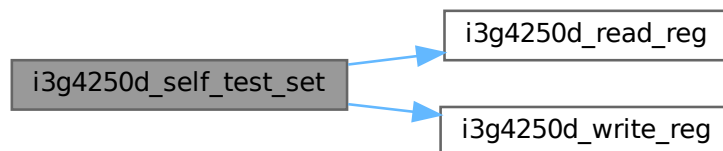
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	change the values of st in reg CTRL_REG4.

Return values

<i>Interface</i>	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

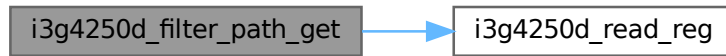
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of out_sel in reg CTRL_REG5.(ptr)

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

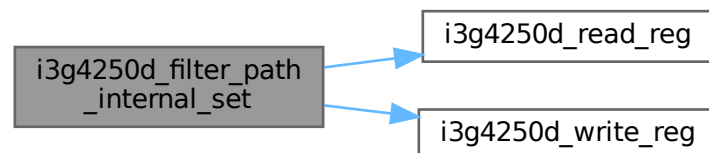
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of int1_sel in reg CTRL_REG5

Return values

<i>Interface</i>	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

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

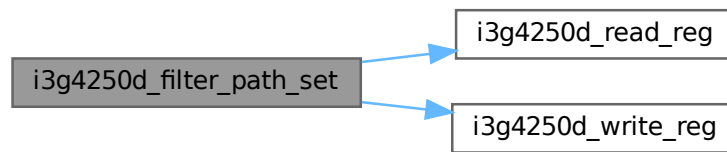
Return values

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

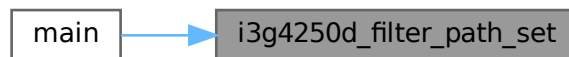
References [i3g4250d_ctrl_reg5_t::hpen](#), [I3G4250D_CTRL_REG5](#), [i3g4250d_read_reg\(\)](#), [i3g4250d_write_reg\(\)](#), and [i3g4250d_ctrl_reg5_t::out_sel](#).

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

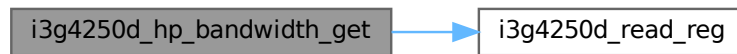
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of hpcf in reg CTRL_REG2.(ptr)

Return values

<i>Interface</i>	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

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

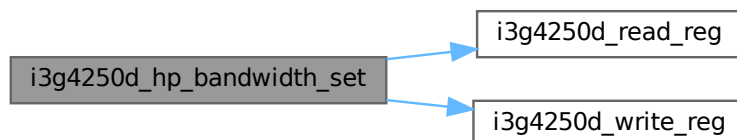
Return values

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

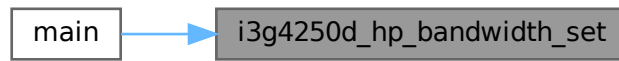
References [i3g4250d_ctrl_reg2_t::hpcf](#), [I3G4250D_CTRL_REG2](#), [i3g4250d_read_reg\(\)](#), and [i3g4250d_write_reg\(\)](#).

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

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

Return values

<i>Interface</i>	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

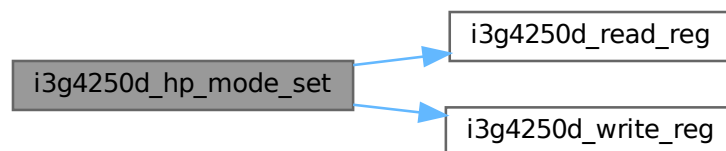
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "hpm" in reg CTRL_REG2.

Return values

<i>Interface</i>	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

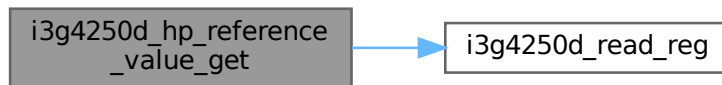
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of ref in reg REFERENCE.(ptr)

Return values

<i>Interface</i>	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

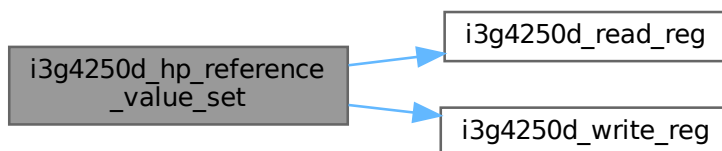
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of ref in reg REFERENCE

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

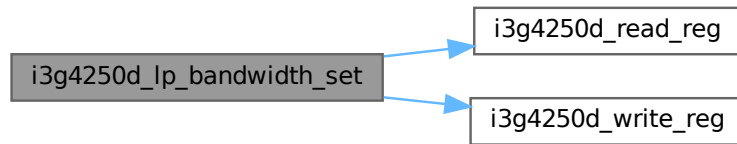
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of "bw" in reg CTRL_REG1.

Return values

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

References [i3g4250d_ctrl_reg1_t::bw](#), [I3G4250D_CTRL_REG1](#), [i3g4250d_read_reg\(\)](#), and [i3g4250d_write_reg\(\)](#).

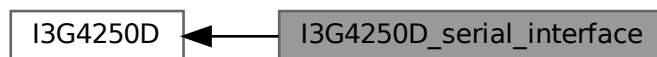
Here is the call graph for this function:



4.1.11 I3G4250D_serial_interface

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

Collaboration diagram for I3G4250D_serial_interface:



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

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

Return values

<i>Interface</i>	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

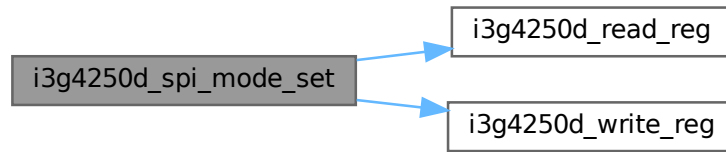
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of sim in reg CTRL_REG4

Return values

<i>Interface</i>	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:



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

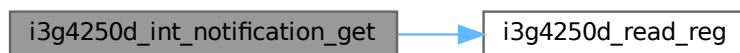
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of lir in reg INT1_CFG.(ptr)

Return values

<i>Interface</i>	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:



4.1.12.2.2 i3g4250d_int_notification_set()

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

Latched/pulsed interrupt.[set].

Parameters

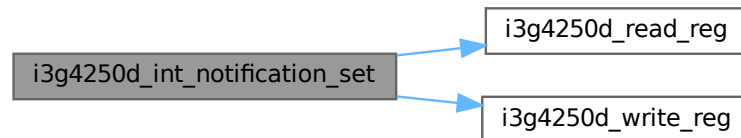
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of lir in reg INT1_CFG.

Return values

<i>Interface</i>	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

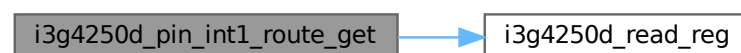
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Read CTRL_REG3 int1 pad.(ptr)

Return values

<i>Interface</i>	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:



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

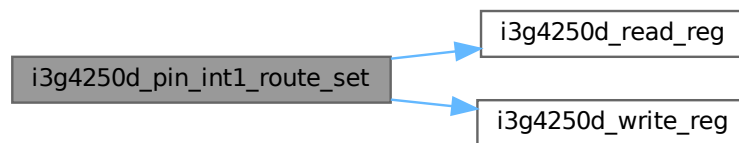
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Configure CTRL_REG3 int1 pad

Return values

<i>Interface</i>	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

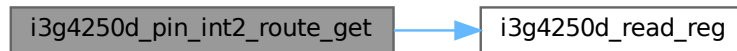
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Read CTRL_REG3 int2 pad.(ptr)

Return values

<i>Interface</i>	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:



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

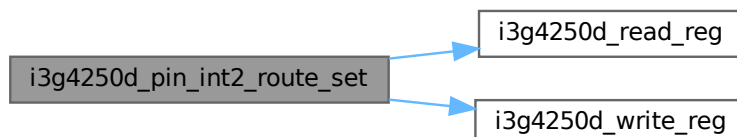
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Configure CTRL_REG3 int2 pad

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

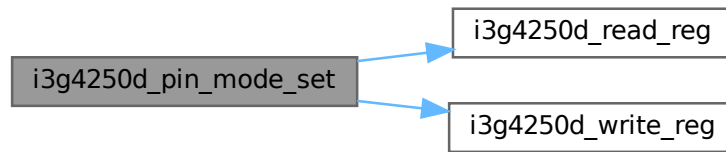
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of pp_od in reg CTRL_REG3

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

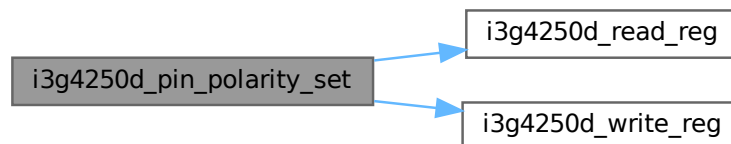
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of h_lactive in reg CTRL_REG3.

Return values

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

References [i3g4250d_ctrl_reg3_t::h_lactive](#), [I3G4250D_CTRL_REG3](#), [i3g4250d_read_reg\(\)](#), and [i3g4250d_write_reg\(\)](#).

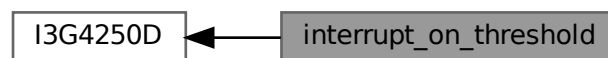
Here is the call graph for this function:



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`:



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

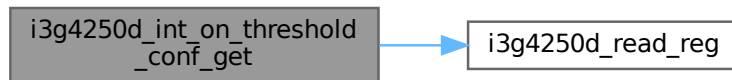
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Struct of registers from INT1_CFG to.(ptr)

Return values

<i>Interface</i>	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

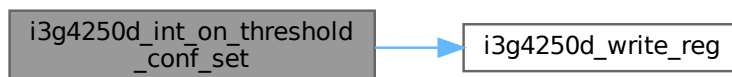
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Struct of registers INT1_CFG

Return values

<i>Interface</i>	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

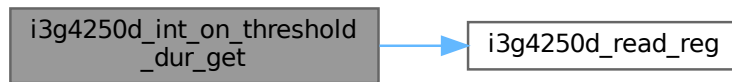
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of d in reg INT1_DURATION.(ptr)

Return values

<i>Interface</i>	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

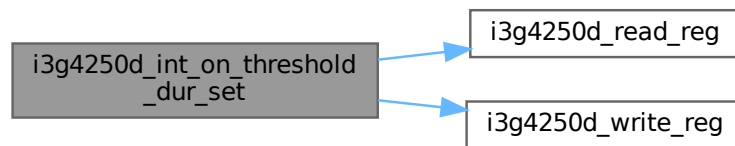
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of d in reg INT1_DURATION

Return values

<i>Interface</i>	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

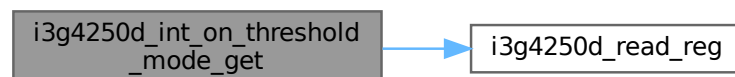
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of and_or in reg INT1_CFG.(ptr)

Return values

<i>Interface</i>	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

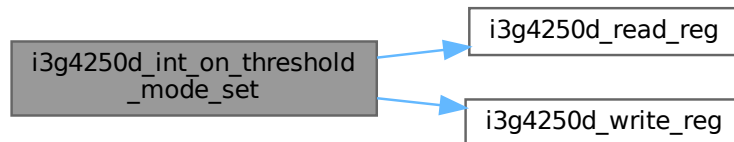
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of and_or in reg INT1_CFG

Return values

<i>Interface</i>	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

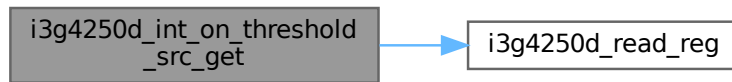
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Union of registers from INT1_SRC to.(ptr)

Return values

<i>Interface</i>	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

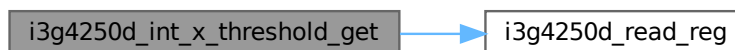
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsx in reg INT1_TSH_XH.(ptr)

Return values

<i>Interface</i>	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

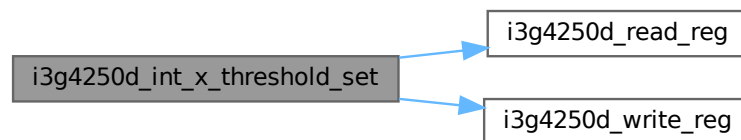
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsx in reg INT1_TSH_XH

Return values

<i>Interface</i>	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

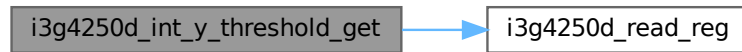
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsy in reg INT1_TSH_YH.(ptr)

Return values

<i>Interface</i>	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

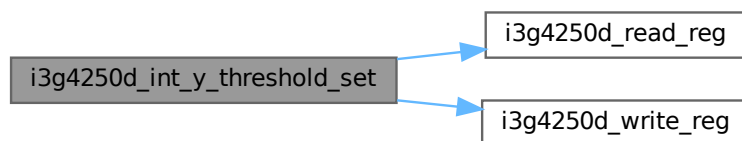
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsy in reg INT1_TSH_YH

Return values

<i>Interface</i>	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

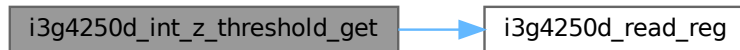
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of thsz in reg INT1_TSH_ZH.(ptr)

Return values

<i>Interface</i>	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

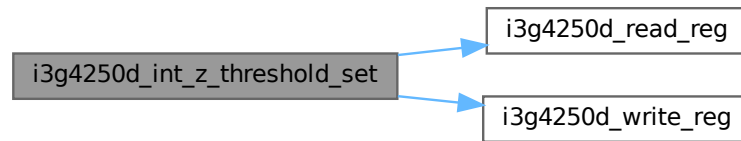
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of thsz in reg INT1_TSH_ZH.

Return values

<i>Interface</i>	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](#).

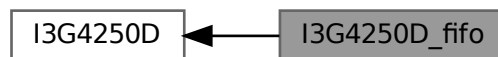
Here is the call graph for this function:



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

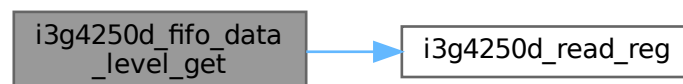
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of fss in reg FIFO_SRC_REG.(ptr)

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

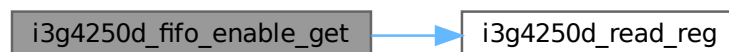
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of fifo_en in reg CTRL_REG5.(ptr)

Return values

<i>Interface</i>	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

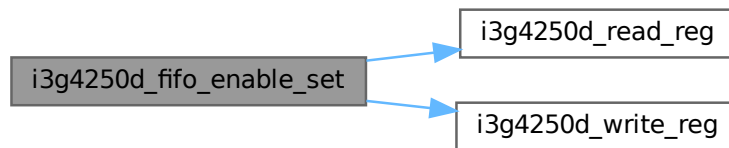
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of fifo_en in reg CTRL_REG5

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

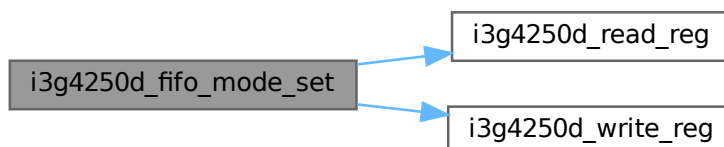
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of fm in reg FIFO_CTRL_REG

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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

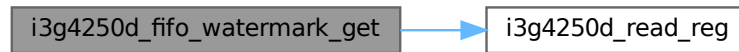
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Get the values of wtm in reg FIFO_CTRL_REG.(ptr)

Return values

<i>Interface</i>	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:



4.1.14.2.9 i3g4250d_fifo_watermark_set()

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

FIFO watermark level selection.[set].

Parameters

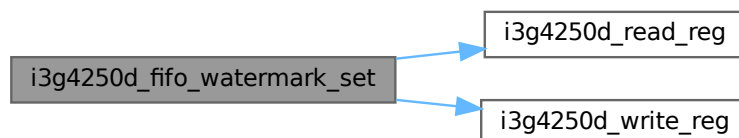
<i>ctx</i>	Read / write interface definitions.(ptr)
<i>val</i>	Change the values of wtm in reg FIFO_CTRL_REG

Return values

<i>Interface</i>	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

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

Return values

<i>Interface</i>	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
```

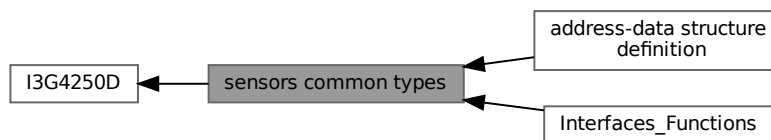
if DRV_BYTE_ORDER is not defined, choose the endianness of your architecture by uncommenting the define which fits your platform endianness

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)
```

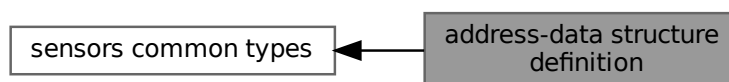
4.1.16.2.4 PROPERTY_ENABLE

```
#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](#)

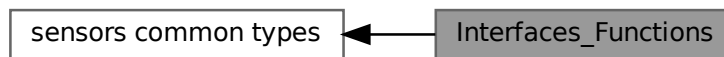
4.1.16.3.1 Detailed Description

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:



Data Structures

- [union i3g4250d_reg_t](#)

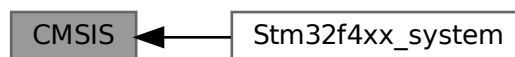
4.1.18.1 Detailed Description

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:



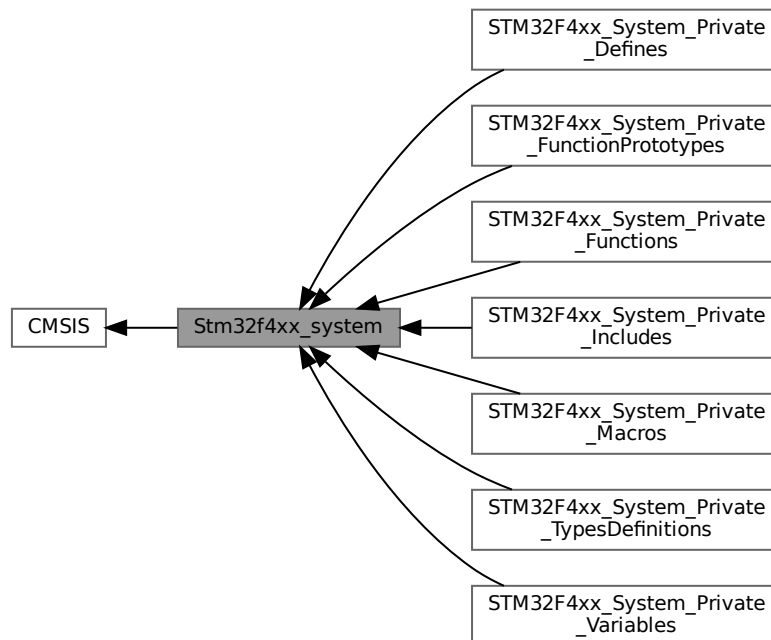
Modules

- [Stm32f4xx_system](#)

4.2.1 Detailed Description

4.2.2 Stm32f4xx_system

Collaboration diagram for Stm32f4xx_system:



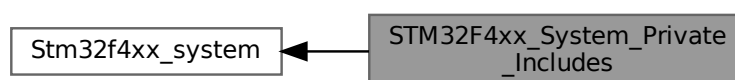
Modules

- [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

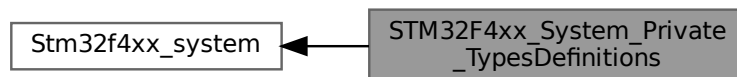
4.2.2.2.2.2 HSI_VALUE

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

Value of the Internal oscillator in Hz

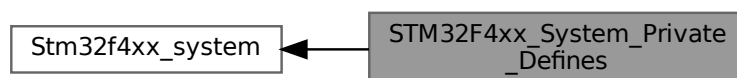
4.2.2.3 STM32F4xx_System_Private_TypeDefinitions

Collaboration diagram for STM32F4xx_System_Private_TypeDefinitions:



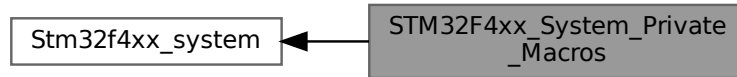
4.2.2.4 STM32F4xx_System_Private_Defines

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}
```

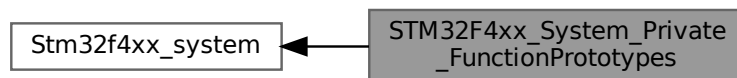

4.2.2.6.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 16000000
```

Referenced by [SystemCoreClockUpdate\(\)](#).

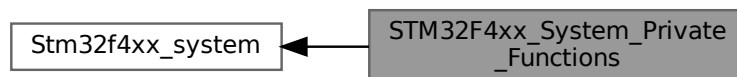
4.2.2.7 STM32F4xx_System_Private_FunctionPrototypes

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 ovrr`: 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_intl_src_t::ia
```

5.12.1.2 not_used_01

```
uint8_t i3g4250d_intl_src_t::not_used_01
```

5.12.1.3 xh

```
uint8_t i3g4250d_intl_src_t::xh
```

5.12.1.4 xl

```
uint8_t i3g4250d_intl_src_t::xl
```

5.12.1.5 yh

```
uint8_t i3g4250d_intl_src_t::yh
```

5.12.1.6 yl

```
uint8_t i3g4250d_intl_src_t::yl
```

5.12.1.7 zh

```
uint8_t i3g4250d_intl_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_y1_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\(\)](#).

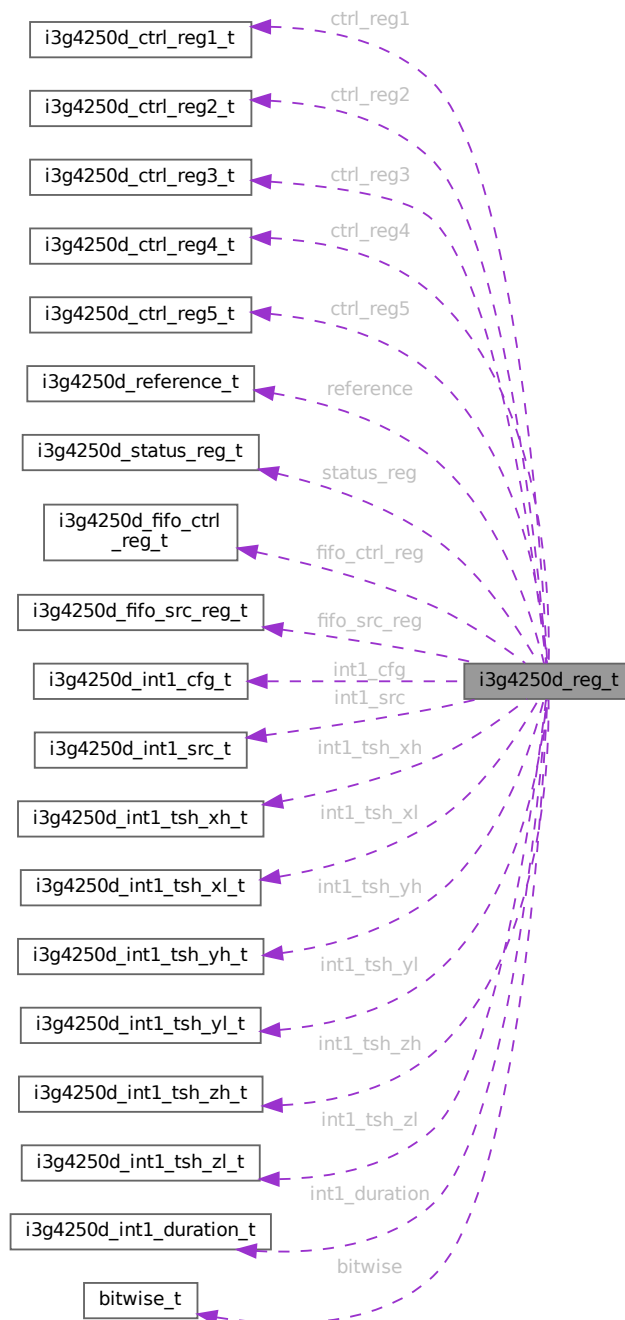
The documentation for this struct was generated from the following file:

- [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

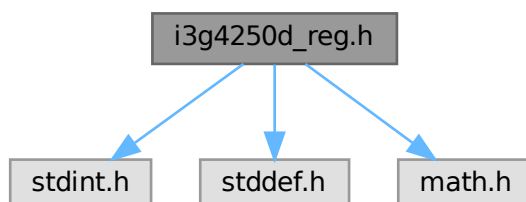
File Documentation

6.1 i3g4250d_reg.h File Reference

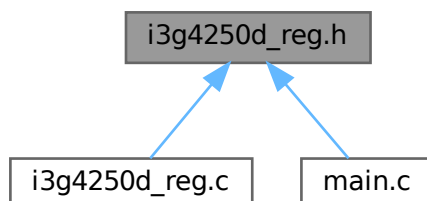
This file contains all the functions prototypes for the [i3g4250d_reg.c](#) driver.

```
#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 aml.[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

© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.2 i3g4250d_reg.h

[Go to the documentation of this file.](#)

```

00001
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef I3G4250D_REGS_H
00023 #define I3G4250D_REGS_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes -----*/
00030 #include <stdint.h>
00031 #include <stddef.h>
00032 #include <math.h>
00033
00044 #ifndef __BYTE_ORDER__
00045
00046 #define DRV_LITTLE_ENDIAN 1234
00047 #define DRV_BIG_ENDIAN 4321
00048
00052 // #define DRV_BYTE_ORDER DRV_BIG_ENDIAN
00053 #define DRV_BYTE_ORDER DRV_LITTLE_ENDIAN
00054
00055 #else /* defined __BYTE_ORDER__ */
00056
00057 #define DRV_LITTLE_ENDIAN __ORDER_LITTLE_ENDIAN__
00058 #define DRV_BIG_ENDIAN __ORDER_BIG_ENDIAN__
00059 #define DRV_BYTE_ORDER __BYTE_ORDER__
00060
00061 #endif /* DRV_BYTE_ORDER */
00062
00073 #ifndef MEMS_SHARED_TYPES
00074 #define MEMS_SHARED_TYPES
00075
00076 typedef struct
00077 {
00078     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00079         uint8_t bit0 : 1;
00080         uint8_t bit1 : 1;
00081         uint8_t bit2 : 1;
00082         uint8_t bit3 : 1;
00083         uint8_t bit4 : 1;
00084         uint8_t bit5 : 1;
00085         uint8_t bit6 : 1;
00086         uint8_t bit7 : 1;
00087     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00088         uint8_t bit7 : 1;
00089         uint8_t bit6 : 1;
00090         uint8_t bit5 : 1;
00091         uint8_t bit4 : 1;
00092         uint8_t bit3 : 1;
00093         uint8_t bit2 : 1;
00094         uint8_t bit1 : 1;
00095         uint8_t bit0 : 1;
00096     #endif /* DRV_BYTE_ORDER */
00097 } bitwise_t;
00098
00099 #define PROPERTY_DISABLE (0U)
00100 #define PROPERTY_ENABLE (1U)
00101
00110 typedef int32_t (*stmdev_write_ptr)(void *, uint8_t, const uint8_t *, uint16_t);
00111 typedef int32_t (*stmdev_read_ptr)(void *, uint8_t, uint8_t *, uint16_t);
00112 typedef void (*stmdev_mdelay_ptr)(uint32_t millisec);
00113
00114 typedef struct
00115 {

```

```

00117     stmdev_write_ptr  write_reg;
00118     stmdev_read_ptr   read_reg;
00120     stmdev_mdelay_ptr mdelay;
00122     void *handle;
00123 } stmdev_ctx_t;
00124
00130 #endif /* MEMS_SHARED_TYPES */
00131
00132 #ifndef MEMS_UCF_SHARED_TYPES
00133 #define MEMS_UCF_SHARED_TYPES
00134
00146 typedef struct
00147 {
00148     uint8_t address;
00149     uint8_t data;
00150 } ucf_line_t;
00151
00157 #endif /* MEMS_UCF_SHARED_TYPES */
00158
00171 #define I3G4250D_I2C_ADD_L      0xD1U
00172 #define I3G4250D_I2C_ADD_H      0xD3U
00173
00175 #define I3G4250D_ID              0xD3U
00176
00182 #define I3G4250D_WHO_AM_I       0x0FU
00183 #define I3G4250D_CTRL_REG1      0x20U
00184 typedef struct
00185 {
00186     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00187         uint8_t pd           : 4; /* xen yen zen pd */
00188         uint8_t bw           : 2;
00189         uint8_t dr           : 2;
00190     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00191         uint8_t dr           : 2;
00192         uint8_t bw           : 2;
00193         uint8_t pd           : 4; /* xen yen zen pd */
00194     #endif /* DRV_BYTE_ORDER */
00195 } i3g4250d_ctrl_reg1_t;
00196
00197 #define I3G4250D_CTRL_REG2      0x21U
00198 typedef struct
00199 {
00200     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00201         uint8_t hpcf         : 4;
00202         uint8_t hpm          : 2;
00203         uint8_t not_used_01  : 2;
00204     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00205         uint8_t not_used_01  : 2;
00206         uint8_t hpm          : 2;
00207         uint8_t hpcf         : 4;
00208     #endif /* DRV_BYTE_ORDER */
00209 } i3g4250d_ctrl_reg2_t;
00210
00211 #define I3G4250D_CTRL_REG3      0x22U
00212 typedef struct
00213 {
00214     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00215         uint8_t i2_empty    : 1;
00216         uint8_t i2_orun     : 1;
00217         uint8_t i2_wtm      : 1;
00218         uint8_t i2_drdy     : 1;
00219         uint8_t pp_od       : 1;
00220         uint8_t h_lactive   : 1;
00221         uint8_t il_boot     : 1;
00222         uint8_t il_int1     : 1;
00223     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00224         uint8_t il_int1     : 1;
00225         uint8_t il_boot     : 1;
00226         uint8_t h_lactive   : 1;
00227         uint8_t pp_od       : 1;
00228         uint8_t i2_drdy     : 1;
00229         uint8_t i2_wtm      : 1;
00230         uint8_t i2_orun     : 1;
00231         uint8_t i2_empty    : 1;
00232     #endif /* DRV_BYTE_ORDER */
00233 } i3g4250d_ctrl_reg3_t;
00234
00235 #define I3G4250D_CTRL_REG4      0x23U
00236 typedef struct
00237 {
00238     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00239         uint8_t sim         : 1;
00240         uint8_t st          : 2;
00241         uint8_t not_used_01 : 1;
00242         uint8_t fs          : 2;
00243         uint8_t ble         : 1;
00244         uint8_t not_used_02 : 1;

```

```

00245 #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00246     uint8_t not_used_02          : 1;
00247     uint8_t ble                  : 1;
00248     uint8_t fs                   : 2;
00249     uint8_t not_used_01         : 1;
00250     uint8_t st                   : 2;
00251     uint8_t sim                  : 1;
00252 #endif /* DRV_BYTE_ORDER */
00253 } i3g4250d_ctrl_reg4_t;
00254
00255 #define I3G4250D_CTRL_REG5          0x24U
00256 typedef struct
00257 {
00258     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00259         uint8_t out_sel          : 2;
00260         uint8_t int1_sel         : 2;
00261         uint8_t hpen             : 1;
00262         uint8_t not_used_01     : 1;
00263         uint8_t fifo_en         : 1;
00264         uint8_t boot            : 1;
00265     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00266         uint8_t boot            : 1;
00267         uint8_t fifo_en         : 1;
00268         uint8_t not_used_01     : 1;
00269         uint8_t hpen            : 1;
00270         uint8_t int1_sel        : 2;
00271         uint8_t out_sel         : 2;
00272     #endif /* DRV_BYTE_ORDER */
00273 } i3g4250d_ctrl_reg5_t;
00274
00275 #define I3G4250D_REFERENCE          0x25U
00276 typedef struct
00277 {
00278     uint8_t ref                  : 8;
00279 } i3g4250d_reference_t;
00280
00281 #define I3G4250D_OUT_TEMP          0x26U
00282 #define I3G4250D_STATUS_REG        0x27U
00283 typedef struct
00284 {
00285     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00286         uint8_t xda              : 1;
00287         uint8_t yda              : 1;
00288         uint8_t zda              : 1;
00289         uint8_t zyxda           : 1;
00290         uint8_t _xor            : 1;
00291         uint8_t yor             : 1;
00292         uint8_t zor             : 1;
00293         uint8_t zyxor           : 1;
00294     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00295         uint8_t zyxor           : 1;
00296         uint8_t zor             : 1;
00297         uint8_t yor             : 1;
00298         uint8_t _xor            : 1;
00299         uint8_t zyxda           : 1;
00300         uint8_t zda             : 1;
00301         uint8_t yda             : 1;
00302         uint8_t xda             : 1;
00303     #endif /* DRV_BYTE_ORDER */
00304 } i3g4250d_status_reg_t;
00305
00306 #define I3G4250D_OUT_X_L           0x28U
00307 #define I3G4250D_OUT_X_H           0x29U
00308 #define I3G4250D_OUT_Y_L           0x2AU
00309 #define I3G4250D_OUT_Y_H           0x2BU
00310 #define I3G4250D_OUT_Z_L           0x2CU
00311 #define I3G4250D_OUT_Z_H           0x2DU
00312 #define I3G4250D_FIFO_CTRL_REG     0x2EU
00313 typedef struct
00314 {
00315     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00316         uint8_t wtm              : 5;
00317         uint8_t fm               : 3;
00318     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00319         uint8_t fm               : 3;
00320         uint8_t wtm              : 5;
00321     #endif /* DRV_BYTE_ORDER */
00322 } i3g4250d_fifo_ctrl_reg_t;
00323
00324 #define I3G4250D_FIFO_SRC_REG      0x2FU
00325 typedef struct
00326 {
00327     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00328         uint8_t fss              : 5;
00329         uint8_t empty            : 1;
00330         uint8_t ovrn             : 1;
00331         uint8_t wtm              : 1;

```

```

00332 #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00333     uint8_t wtm : 1;
00334     uint8_t ovrn : 1;
00335     uint8_t empty : 1;
00336     uint8_t fss : 5;
00337 #endif /* DRV_BYTE_ORDER */
00338 } i3g4250d_fifo_src_reg_t;
00339
00340 #define I3G4250D_INT1_CFG 0x30U
00341 typedef struct
00342 {
00343     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00344         uint8_t xlie : 1;
00345         uint8_t xhie : 1;
00346         uint8_t ylie : 1;
00347         uint8_t yhie : 1;
00348         uint8_t zlie : 1;
00349         uint8_t zhie : 1;
00350         uint8_t lir : 1;
00351         uint8_t and_or : 1;
00352     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00353         uint8_t and_or : 1;
00354         uint8_t lir : 1;
00355         uint8_t zhie : 1;
00356         uint8_t zlie : 1;
00357         uint8_t yhie : 1;
00358         uint8_t ylie : 1;
00359         uint8_t xhie : 1;
00360         uint8_t xlie : 1;
00361     #endif /* DRV_BYTE_ORDER */
00362 } i3g4250d_int1_cfg_t;
00363
00364 #define I3G4250D_INT1_SRC 0x31U
00365 typedef struct
00366 {
00367     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00368         uint8_t xl : 1;
00369         uint8_t xh : 1;
00370         uint8_t yl : 1;
00371         uint8_t yh : 1;
00372         uint8_t zl : 1;
00373         uint8_t zh : 1;
00374         uint8_t ia : 1;
00375         uint8_t not_used_01 : 1;
00376     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00377         uint8_t not_used_01 : 1;
00378         uint8_t ia : 1;
00379         uint8_t zh : 1;
00380         uint8_t zl : 1;
00381         uint8_t yh : 1;
00382         uint8_t yl : 1;
00383         uint8_t xh : 1;
00384         uint8_t xl : 1;
00385     #endif /* DRV_BYTE_ORDER */
00386 } i3g4250d_int1_src_t;
00387
00388 #define I3G4250D_INT1_TSH_XH 0x32U
00389 typedef struct
00390 {
00391     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00392         uint8_t thsx : 7;
00393         uint8_t not_used_01 : 1;
00394     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00395         uint8_t not_used_01 : 1;
00396         uint8_t thsx : 7;
00397     #endif /* DRV_BYTE_ORDER */
00398 } i3g4250d_int1_tsh_xh_t;
00399
00400 #define I3G4250D_INT1_TSH_XL 0x33U
00401 typedef struct
00402 {
00403     uint8_t thsx : 8;
00404 } i3g4250d_int1_tsh_xl_t;
00405
00406 #define I3G4250D_INT1_TSH_YH 0x34U
00407 typedef struct
00408 {
00409     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00410         uint8_t thsy : 7;
00411         uint8_t not_used_01 : 1;
00412     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00413         uint8_t not_used_01 : 1;
00414         uint8_t thsy : 7;
00415     #endif /* DRV_BYTE_ORDER */
00416 } i3g4250d_int1_tsh_yh_t;
00417
00418 #define I3G4250D_INT1_TSH_YL 0x35U

```

```

00419 typedef struct
00420 {
00421     uint8_t thsy                : 8;
00422 } i3g4250d_int1_tsh_y1_t;
00423
00424 #define I3G4250D_INT1_TSH_ZH          0x36U
00425 typedef struct
00426 {
00427     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00428         uint8_t thsz            : 7;
00429         uint8_t not_used_01     : 1;
00430     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00431         uint8_t not_used_01     : 1;
00432         uint8_t thsz            : 7;
00433     #endif /* DRV_BYTE_ORDER */
00434 } i3g4250d_int1_tsh_zh_t;
00435
00436 #define I3G4250D_INT1_TSH_ZL          0x37U
00437 typedef struct
00438 {
00439     uint8_t thsz                : 8;
00440 } i3g4250d_int1_tsh_zl_t;
00441
00442 #define I3G4250D_INT1_DURATION        0x38U
00443 typedef struct
00444 {
00445     #if DRV_BYTE_ORDER == DRV_LITTLE_ENDIAN
00446         uint8_t d                : 7;
00447         uint8_t wait             : 1;
00448     #elif DRV_BYTE_ORDER == DRV_BIG_ENDIAN
00449         uint8_t wait             : 1;
00450         uint8_t d                : 7;
00451     #endif /* DRV_BYTE_ORDER */
00452 } i3g4250d_int1_duration_t;
00453
00454 typedef union
00455 {
00456     i3g4250d_ctrl_reg1_t        ctrl_reg1;
00457     i3g4250d_ctrl_reg2_t        ctrl_reg2;
00458     i3g4250d_ctrl_reg3_t        ctrl_reg3;
00459     i3g4250d_ctrl_reg4_t        ctrl_reg4;
00460     i3g4250d_ctrl_reg5_t        ctrl_reg5;
00461     i3g4250d_reference_t        reference;
00462     i3g4250d_status_reg_t       status_reg;
00463     i3g4250d_fifo_ctrl_reg_t    fifo_ctrl_reg;
00464     i3g4250d_fifo_src_reg_t     fifo_src_reg;
00465     i3g4250d_int1_cfg_t         int1_cfg;
00466     i3g4250d_int1_src_t         int1_src;
00467     i3g4250d_int1_tsh_xh_t      int1_tsh_xh;
00468     i3g4250d_int1_tsh_xl_t      int1_tsh_xl;
00469     i3g4250d_int1_tsh_yh_t      int1_tsh_yh;
00470     i3g4250d_int1_tsh_yl_t      int1_tsh_yl;
00471     i3g4250d_int1_tsh_zh_t      int1_tsh_zh;
00472     i3g4250d_int1_tsh_zl_t      int1_tsh_zl;
00473     i3g4250d_int1_duration_t    int1_duration;
00474     bitwise_t                   bitwise;
00475     uint8_t                     byte;
00476 } i3g4250d_reg_t;
00477
00478 #ifndef __weak
00479 #define __weak __attribute__((weak))
00480 #endif /* __weak */
00481
00482 /*
00483  * These are the basic platform dependent I/O routines to read
00484  * and write device registers connected on a standard bus.
00485  * The driver keeps offering a default implementation based on function
00486  * pointers to read/write routines for backward compatibility.
00487  * The __weak directive allows the final application to overwrite
00488  * them with a custom implementation.
00489  */
00490
00491 int32_t i3g4250d_read_reg(const stmdev_ctx_t *ctx, uint8_t reg,
00492                          uint8_t *data,
00493                          uint16_t len);
00494 int32_t i3g4250d_write_reg(const stmdev_ctx_t *ctx, uint8_t reg,
00495                            uint8_t *data,
00496                            uint16_t len);
00497
00498 float_t i3g4250d_from_fs245dps_to_mdps(int16_t lsb);
00499 float_t i3g4250d_from_lsb_to_celsius(int16_t lsb);
00500
00501 int32_t i3g4250d_axis_x_data_set(const stmdev_ctx_t *ctx, uint8_t val);
00502 int32_t i3g4250d_axis_x_data_get(const stmdev_ctx_t *ctx, uint8_t *val);
00503
00504 int32_t i3g4250d_axis_y_data_set(const stmdev_ctx_t *ctx, uint8_t val);
00505 int32_t i3g4250d_axis_y_data_get(const stmdev_ctx_t *ctx, uint8_t *val);

```

```

00524
00525 int32_t i3g4250d_axis_z_data_set(const stmdev_ctx_t *ctx, uint8_t val);
00526 int32_t i3g4250d_axis_z_data_get(const stmdev_ctx_t *ctx, uint8_t *val);
00527
00528 typedef enum
00529 {
00530     I3G4250D_ODR_OFF      = 0x00,
00531     I3G4250D_ODR_SLEEP   = 0x08,
00532     I3G4250D_ODR_100Hz   = 0x0F,
00533     I3G4250D_ODR_200Hz   = 0x1F,
00534     I3G4250D_ODR_400Hz   = 0x2F,
00535     I3G4250D_ODR_800Hz   = 0x3F,
00536 } i3g4250d_dr_t;
00537 int32_t i3g4250d_data_rate_set(const stmdev_ctx_t *ctx, i3g4250d_dr_t val);
00538 int32_t i3g4250d_data_rate_get(const stmdev_ctx_t *ctx, i3g4250d_dr_t *val);
00539
00540 typedef enum
00541 {
00542     I3G4250D_245dps       = 0x00,
00543     I3G4250D_500dps       = 0x01,
00544     I3G4250D_2000dps      = 0x02,
00545 } i3g4250d_fs_t;
00546 int32_t i3g4250d_full_scale_set(const stmdev_ctx_t *ctx, i3g4250d_fs_t val);
00547 int32_t i3g4250d_full_scale_get(const stmdev_ctx_t *ctx,
00548                                 i3g4250d_fs_t *val);
00549
00550 int32_t i3g4250d_status_reg_get(const stmdev_ctx_t *ctx,
00551                                 i3g4250d_status_reg_t *val);
00552
00553 int32_t i3g4250d_flag_data_ready_get(const stmdev_ctx_t *ctx, uint8_t *val);
00554
00555 int32_t i3g4250d_temperature_raw_get(const stmdev_ctx_t *ctx,
00556                                       uint8_t *buff);
00557
00558 int32_t i3g4250d_angular_rate_raw_get(const stmdev_ctx_t *ctx,
00559                                       int16_t *val);
00560
00561 int32_t i3g4250d_device_id_get(const stmdev_ctx_t *ctx, uint8_t *buff);
00562
00563 typedef enum
00564 {
00565     I3G4250D_GY_ST_DISABLE = 0,
00566     I3G4250D_GY_ST_POSITIVE = 1,
00567     I3G4250D_GY_ST_NEGATIVE = 3,
00568 } i3g4250d_st_t;
00569 int32_t i3g4250d_self_test_set(const stmdev_ctx_t *ctx, i3g4250d_st_t val);
00570 int32_t i3g4250d_self_test_get(const stmdev_ctx_t *ctx, i3g4250d_st_t *val);
00571
00572 typedef enum
00573 {
00574     I3G4250D_AUX_LSB_AT_LOW_ADD = 0,
00575     I3G4250D_AUX_MSB_AT_LOW_ADD = 1,
00576 } i3g4250d_ble_t;
00577 int32_t i3g4250d_data_format_set(const stmdev_ctx_t *ctx,
00578                                   i3g4250d_ble_t val);
00579 int32_t i3g4250d_data_format_get(const stmdev_ctx_t *ctx,
00580                                   i3g4250d_ble_t *val);
00581
00582 int32_t i3g4250d_boot_set(const stmdev_ctx_t *ctx, uint8_t val);
00583 int32_t i3g4250d_boot_get(const stmdev_ctx_t *ctx, uint8_t *val);
00584
00585 typedef enum
00586 {
00587     I3G4250D_CUT_OFF_LOW      = 0,
00588     I3G4250D_CUT_OFF_MEDIUM   = 1,
00589     I3G4250D_CUT_OFF_HIGH     = 2,
00590     I3G4250D_CUT_OFF_VERY_HIGH = 3,
00591 } i3g4250d_bw_t;
00592 int32_t i3g4250d_lp_bandwidth_set(const stmdev_ctx_t *ctx,
00593                                    i3g4250d_bw_t val);
00594 int32_t i3g4250d_lp_bandwidth_get(const stmdev_ctx_t *ctx,
00595                                    i3g4250d_bw_t *val);
00596
00597 typedef enum
00598 {
00599     I3G4250D_HP_LEVEL_0 = 0,
00600     I3G4250D_HP_LEVEL_1 = 1,
00601     I3G4250D_HP_LEVEL_2 = 2,
00602     I3G4250D_HP_LEVEL_3 = 3,
00603     I3G4250D_HP_LEVEL_4 = 4,
00604     I3G4250D_HP_LEVEL_5 = 5,
00605     I3G4250D_HP_LEVEL_6 = 6,
00606     I3G4250D_HP_LEVEL_7 = 7,
00607     I3G4250D_HP_LEVEL_8 = 8,
00608     I3G4250D_HP_LEVEL_9 = 9,
00609 } i3g4250d_hpcf_t;
00610 int32_t i3g4250d_hp_bandwidth_set(const stmdev_ctx_t *ctx,

```



```

00611                                     i3g4250d_hpcf_t val);
00612 int32_t i3g4250d_hp_bandwidth_get(const stmdev_ctx_t *ctx,
00613                                     i3g4250d_hpcf_t *val);
00614
00615 typedef enum
00616 {
00617     I3G4250D_HP_NORMAL_MODE_WITH_RST = 0,
00618     I3G4250D_HP_REFERENCE_SIGNAL     = 1,
00619     I3G4250D_HP_NORMAL_MODE         = 2,
00620     I3G4250D_HP_AUTO_RESET_ON_INT   = 3,
00621 } i3g4250d_hpm_t;
00622 int32_t i3g4250d_hp_mode_set(const stmdev_ctx_t *ctx, i3g4250d_hpm_t val);
00623 int32_t i3g4250d_hp_mode_get(const stmdev_ctx_t *ctx, i3g4250d_hpm_t *val);
00624
00625 typedef enum
00626 {
00627     I3G4250D_ONLY_LPF1_ON_OUT        = 0,
00628     I3G4250D_LPF1_HP_ON_OUT         = 1,
00629     I3G4250D_LPF1_LPF2_ON_OUT       = 2,
00630     I3G4250D_LPF1_HP_LPF2_ON_OUT    = 6,
00631 } i3g4250d_out_sel_t;
00632 int32_t i3g4250d_filter_path_set(const stmdev_ctx_t *ctx,
00633                                     i3g4250d_out_sel_t val);
00634 int32_t i3g4250d_filter_path_get(const stmdev_ctx_t *ctx,
00635                                     i3g4250d_out_sel_t *val);
00636
00637 typedef enum
00638 {
00639     I3G4250D_ONLY_LPF1_ON_INT        = 0,
00640     I3G4250D_LPF1_HP_ON_INT         = 1,
00641     I3G4250D_LPF1_LPF2_ON_INT       = 2,
00642     I3G4250D_LPF1_HP_LPF2_ON_INT    = 6,
00643 } i3g4250d_int1_sel_t;
00644 int32_t i3g4250d_filter_path_internal_set(const stmdev_ctx_t *ctx,
00645                                             i3g4250d_int1_sel_t val);
00646 int32_t i3g4250d_filter_path_internal_get(const stmdev_ctx_t *ctx,
00647                                             i3g4250d_int1_sel_t *val);
00648
00649 int32_t i3g4250d_hp_reference_value_set(const stmdev_ctx_t *ctx,
00650                                         uint8_t val);
00651 int32_t i3g4250d_hp_reference_value_get(const stmdev_ctx_t *ctx,
00652                                         uint8_t *val);
00653
00654 typedef enum
00655 {
00656     I3G4250D_SPI_4_WIRE = 0,
00657     I3G4250D_SPI_3_WIRE = 1,
00658 } i3g4250d_sim_t;
00659 int32_t i3g4250d_spi_mode_set(const stmdev_ctx_t *ctx, i3g4250d_sim_t val);
00660 int32_t i3g4250d_spi_mode_get(const stmdev_ctx_t *ctx, i3g4250d_sim_t *val);
00661
00662 typedef struct
00663 {
00664     uint8_t i1_int1          : 1;
00665     uint8_t i1_boot          : 1;
00666 } i3g4250d_int1_route_t;
00667 int32_t i3g4250d_pin_int1_route_set(const stmdev_ctx_t *ctx,
00668                                     i3g4250d_int1_route_t val);
00669 int32_t i3g4250d_pin_int1_route_get(const stmdev_ctx_t *ctx,
00670                                     i3g4250d_int1_route_t *val);
00671
00672 typedef struct
00673 {
00674     uint8_t i2_empty          : 1;
00675     uint8_t i2_orun           : 1;
00676     uint8_t i2_wtm            : 1;
00677     uint8_t i2_drdy           : 1;
00678 } i3g4250d_int2_route_t;
00679 int32_t i3g4250d_pin_int2_route_set(const stmdev_ctx_t *ctx,
00680                                     i3g4250d_int2_route_t val);
00681 int32_t i3g4250d_pin_int2_route_get(const stmdev_ctx_t *ctx,
00682                                     i3g4250d_int2_route_t *val);
00683
00684 typedef enum
00685 {
00686     I3G4250D_PUSH_PULL = 0,
00687     I3G4250D_OPEN_DRAIN = 1,
00688 } i3g4250d_pp_od_t;
00689 int32_t i3g4250d_pin_mode_set(const stmdev_ctx_t *ctx,
00690                               i3g4250d_pp_od_t val);
00691 int32_t i3g4250d_pin_mode_get(const stmdev_ctx_t *ctx,
00692                               i3g4250d_pp_od_t *val);
00693
00694 typedef enum
00695 {
00696     I3G4250D_ACTIVE_HIGH = 0,
00697     I3G4250D_ACTIVE_LOW  = 1,

```

```

00698 } i3g4250d_h_lactive_t;
00699 int32_t i3g4250d_pin_polarity_set(const stmdev_ctx_t *ctx,
00700                                i3g4250d_h_lactive_t val);
00701 int32_t i3g4250d_pin_polarity_get(const stmdev_ctx_t *ctx,
00702                                i3g4250d_h_lactive_t *val);
00703
00704 typedef enum
00705 {
00706     I3G4250D_INT_PULSED    = 0,
00707     I3G4250D_INT_LATCHED  = 1,
00708 } i3g4250d_lir_t;
00709 int32_t i3g4250d_int_notification_set(const stmdev_ctx_t *ctx,
00710                                     i3g4250d_lir_t val);
00711 int32_t i3g4250d_int_notification_get(const stmdev_ctx_t *ctx,
00712                                     i3g4250d_lir_t *val);
00713
00714 int32_t i3g4250d_int_on_threshold_conf_set(const stmdev_ctx_t *ctx,
00715                                           i3g4250d_int1_cfg_t *val);
00716 int32_t i3g4250d_int_on_threshold_conf_get(const stmdev_ctx_t *ctx,
00717                                           i3g4250d_int1_cfg_t *val);
00718
00719 typedef enum
00720 {
00721     I3G4250D_INT1_ON_TH_AND = 1,
00722     I3G4250D_INT1_ON_TH_OR  = 0,
00723 } i3g4250d_and_or_t;
00724 int32_t i3g4250d_int_on_threshold_mode_set(const stmdev_ctx_t *ctx,
00725                                           i3g4250d_and_or_t val);
00726 int32_t i3g4250d_int_on_threshold_mode_get(const stmdev_ctx_t *ctx,
00727                                           i3g4250d_and_or_t *val);
00728
00729 int32_t i3g4250d_int_on_threshold_src_get(const stmdev_ctx_t *ctx,
00730                                           i3g4250d_int1_src_t *val);
00731
00732 int32_t i3g4250d_int_x_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
00733 int32_t i3g4250d_int_x_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);
00734
00735 int32_t i3g4250d_int_y_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
00736 int32_t i3g4250d_int_y_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);
00737
00738 int32_t i3g4250d_int_z_threshold_set(const stmdev_ctx_t *ctx, uint16_t val);
00739 int32_t i3g4250d_int_z_threshold_get(const stmdev_ctx_t *ctx, uint16_t *val);
00740
00741 int32_t i3g4250d_int_on_threshold_dur_set(const stmdev_ctx_t *ctx,
00742                                           uint8_t val);
00743 int32_t i3g4250d_int_on_threshold_dur_get(const stmdev_ctx_t *ctx,
00744                                           uint8_t *val);
00745
00746 int32_t i3g4250d_fifo_enable_set(const stmdev_ctx_t *ctx, uint8_t val);
00747 int32_t i3g4250d_fifo_enable_get(const stmdev_ctx_t *ctx, uint8_t *val);
00748
00749 int32_t i3g4250d_fifo_watermark_set(const stmdev_ctx_t *ctx, uint8_t val);
00750 int32_t i3g4250d_fifo_watermark_get(const stmdev_ctx_t *ctx, uint8_t *val);
00751
00752 typedef enum
00753 {
00754     I3G4250D_FIFO_BYPASS_MODE    = 0x00,
00755     I3G4250D_FIFO_MODE           = 0x01,
00756     I3G4250D_FIFO_STREAM_MODE    = 0x02,
00757 } i3g4250d_fifo_mode_t;
00758 int32_t i3g4250d_fifo_mode_set(const stmdev_ctx_t *ctx,
00759                               i3g4250d_fifo_mode_t val);
00760 int32_t i3g4250d_fifo_mode_get(const stmdev_ctx_t *ctx,
00761                               i3g4250d_fifo_mode_t *val);
00762
00763 int32_t i3g4250d_fifo_data_level_get(const stmdev_ctx_t *ctx, uint8_t *val);
00764
00765 int32_t i3g4250d_fifo_empty_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);
00766
00767 int32_t i3g4250d_fifo_ovr_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);
00768
00769 int32_t i3g4250d_fifo_wtm_flag_get(const stmdev_ctx_t *ctx, uint8_t *val);
00770
00771 #ifdef __cplusplus
00772 }
00773 #endif
00774
00775 #endif /* I3G4250D_REGS_H */
00776
00777 /***** (C) COPYRIGHT STMicroelectronics *****/

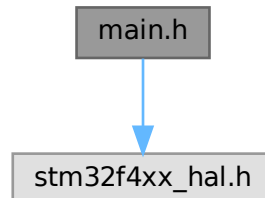
```

6.3 main.h File Reference

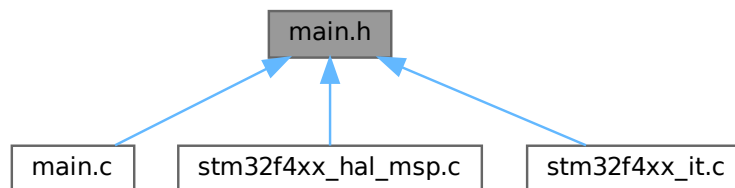
: Header for [main.c](#) file. This file contains the common defines of the application.

```
#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

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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.](#)

```
00001 /* USER CODE BEGIN Header */
00019 /* USER CODE END Header */
00020
00021 /* Define to prevent recursive inclusion -----*/
00022 #ifndef __MAIN_H
00023 #define __MAIN_H
00024
00025 #ifdef __cplusplus
00026 extern "C" {
00027 #endif
00028
00029 /* Includes -----*/
00030 #include "stm32f4xx_hal.h"
00031
00032 /* Private includes -----*/
00033 /* USER CODE BEGIN Includes */
00034
00035 /* USER CODE END Includes */
00036
00037 /* Exported types -----*/
00038 /* USER CODE BEGIN ET */
00039
00040 /* USER CODE END ET */
00041
```

```

00042 /* Exported constants -----*/
00043 /* USER CODE BEGIN EC */
00044
00045 /* USER CODE END EC */
00046
00047 /* Exported macro -----*/
00048 /* USER CODE BEGIN EM */
00049
00050 /* USER CODE END EM */
00051
00052 /* Exported functions prototypes -----*/
00053 void Error_Handler(void);
00054
00055 /* USER CODE BEGIN EFP */
00056
00057 /* USER CODE END EFP */
00058
00059 /* Private defines -----*/
00060 #define GYRO_CS_OUT_Pin GPIO_PIN_4
00061 #define GYRO_CS_OUT_GPIO_Port GPIOA
00062
00063 /* USER CODE BEGIN Private defines */
00064
00065 /* USER CODE END Private defines */
00066
00067 #ifdef __cplusplus
00068 }
00069 #endif
00070
00071 #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 0x0000FFFU
- #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 0x0000FFFU
- #define PHY_RESET ((uint16_t)0x8000U)
- #define PHY_RESET_DELAY 0x00000FFU
- #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
- #define PHY_SPEED_STATUS ((uint16_t))
- #define PHY_SR ((uint16_t))
- #define PHY_WRITE_TO 0x0000FFFU
- #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_FMPMBUS_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_CRYPT_REGISTER_CALLBACKS

```
#define USE_HAL_CRYPT_REGISTER_CALLBACKS 0U /* CRYPT 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.](#)

```
00001 /* USER CODE BEGIN Header */
00021 /* USER CODE END Header */
00022
00023 /* Define to prevent recursive inclusion -----*/
00024 #ifndef __STM32F4xx_HAL_CONF_H
00025 #define __STM32F4xx_HAL_CONF_H
00026
00027 #ifdef __cplusplus
00028     extern "C" {
00029 #endif
00030
00031 /* Exported types -----*/
00032 /* Exported constants -----*/
00033
00034 /* ##### Module Selection ##### */
00038 #define HAL_MODULE_ENABLED
00039
00040 /* #define HAL_Cryp_MODULE_ENABLED */
00041 /* #define HAL_ADC_MODULE_ENABLED */
00042 /* #define HAL_CAN_MODULE_ENABLED */
00043 /* #define HAL_CRC_MODULE_ENABLED */
00044 /* #define HAL_CAN_LEGACY_MODULE_ENABLED */
00045 /* #define HAL_DAC_MODULE_ENABLED */
00046 /* #define HAL_DCMI_MODULE_ENABLED */
00047 /* #define HAL_DMA2D_MODULE_ENABLED */
00048 /* #define HAL_ETH_MODULE_ENABLED */
00049 /* #define HAL_ETH_LEGACY_MODULE_ENABLED */
00050 /* #define HAL_NAND_MODULE_ENABLED */
00051 /* #define HAL_NOR_MODULE_ENABLED */
00052 /* #define HAL_PCCARD_MODULE_ENABLED */
00053 /* #define HAL_SRAM_MODULE_ENABLED */
00054 /* #define HAL_SDRAM_MODULE_ENABLED */
00055 /* #define HAL_HASH_MODULE_ENABLED */
00056 /* #define HAL_I2C_MODULE_ENABLED */
00057 /* #define HAL_I2S_MODULE_ENABLED */
00058 /* #define HAL_IWDG_MODULE_ENABLED */
00059 /* #define HAL_LTDC_MODULE_ENABLED */
00060 /* #define HAL_RNG_MODULE_ENABLED */
00061 /* #define HAL_RTC_MODULE_ENABLED */
00062 /* #define HAL_SAI_MODULE_ENABLED */
00063 /* #define HAL_SD_MODULE_ENABLED */
00064 /* #define HAL_MMC_MODULE_ENABLED */
00065 #define HAL_SPI_MODULE_ENABLED
00066 /* #define HAL_TIM_MODULE_ENABLED */
00067 #define HAL_UART_MODULE_ENABLED
00068 /* #define HAL_USART_MODULE_ENABLED */
00069 /* #define HAL_IRDA_MODULE_ENABLED */
00070 /* #define HAL_SMARTCARD_MODULE_ENABLED */
00071 /* #define HAL_SMBUS_MODULE_ENABLED */
00072 /* #define HAL_WWDG_MODULE_ENABLED */
00073 /* #define HAL_PCD_MODULE_ENABLED */
00074 /* #define HAL_HCD_MODULE_ENABLED */
00075 /* #define HAL_DSI_MODULE_ENABLED */
00076 /* #define HAL_QSPI_MODULE_ENABLED */
00077 /* #define HAL_QSPI_MODULE_ENABLED */
00078 /* #define HAL_CEC_MODULE_ENABLED */
00079 /* #define HAL_FMPI2C_MODULE_ENABLED */
00080 /* #define HAL_FMPMBUS_MODULE_ENABLED */
00081 /* #define HAL_SPDIFRX_MODULE_ENABLED */
00082 /* #define HAL_DFSDM_MODULE_ENABLED */
00083 /* #define HAL_LPTIM_MODULE_ENABLED */
00084 #define HAL_GPIO_MODULE_ENABLED
00085 #define HAL_EXTI_MODULE_ENABLED
00086 #define HAL_DMA_MODULE_ENABLED
00087 #define HAL_RCC_MODULE_ENABLED
00088 #define HAL_FLASH_MODULE_ENABLED
```

```
00089 #define HAL_PWR_MODULE_ENABLED
00090 #define HAL_CORTEX_MODULE_ENABLED
00091
00092 /* ##### HSE/HSI Values adaptation ##### */
00093 #if !defined (HSE_VALUE)
00094 #define HSE_VALUE 25000000U
00095 #endif /* HSE_VALUE */
00096
00097 #if !defined (HSE_STARTUP_TIMEOUT)
00098 #define HSE_STARTUP_TIMEOUT 100U
00099 #endif /* HSE_STARTUP_TIMEOUT */
00100
00101 #if !defined (HSI_VALUE)
00102 #define HSI_VALUE ((uint32_t)16000000U)
00103 #endif /* HSI_VALUE */
00104
00105 #if !defined (LSI_VALUE)
00106 #define LSI_VALUE 32000U
00107 #endif /* LSI_VALUE */
00108
00109 #if !defined (LSE_VALUE)
00110 #define LSE_VALUE 32768U
00111 #endif /* LSE_VALUE */
00112
00113 #if !defined (LSE_STARTUP_TIMEOUT)
00114 #define LSE_STARTUP_TIMEOUT 5000U
00115 #endif /* LSE_STARTUP_TIMEOUT */
00116
00117 #if !defined (EXTERNAL_CLOCK_VALUE)
00118 #define EXTERNAL_CLOCK_VALUE 12288000U
00119 #endif /* EXTERNAL_CLOCK_VALUE */
00120
00121 /* Tip: To avoid modifying this file each time you need to use different HSE,
00122 == you can define the HSE value in your toolchain compiler preprocessor. */
00123
00124 /* ##### System Configuration ##### */
00125 #define VDD_VALUE 3300U
00126 #define TICK_INT_PRIORITY 15U
00127 #define USE_RTOS 0U
00128 #define PREFETCH_ENABLE 1U
00129 #define INSTRUCTION_CACHE_ENABLE 1U
00130 #define DATA_CACHE_ENABLE 1U
00131
00132 #define USE_HAL_ADC_REGISTER_CALLBACKS 0U /* ADC register callback disabled */
00133 #define USE_HAL_CAN_REGISTER_CALLBACKS 0U /* CAN register callback disabled */
00134 #define USE_HAL_CEC_REGISTER_CALLBACKS 0U /* CEC register callback disabled */
00135 #define USE_HAL_Cryp_REGISTER_CALLBACKS 0U /* CRYP register callback disabled */
00136 #define USE_HAL_DAC_REGISTER_CALLBACKS 0U /* DAC register callback disabled */
00137 #define USE_HAL_DCMI_REGISTER_CALLBACKS 0U /* DCMI register callback disabled */
00138 #define USE_HAL_DFSDM_REGISTER_CALLBACKS 0U /* DFSDM register callback disabled */
00139 #define USE_HAL_DMA2D_REGISTER_CALLBACKS 0U /* DMA2D register callback disabled */
00140 #define USE_HAL_DSI_REGISTER_CALLBACKS 0U /* DSI register callback disabled */
00141 #define USE_HAL_ETH_REGISTER_CALLBACKS 0U /* ETH register callback disabled */
00142 #define USE_HAL_HASH_REGISTER_CALLBACKS 0U /* HASH register callback disabled */
00143 #define USE_HAL_HCD_REGISTER_CALLBACKS 0U /* HCD register callback disabled */
00144 #define USE_HAL_I2C_REGISTER_CALLBACKS 0U /* I2C register callback disabled */
00145 #define USE_HAL_FMPI2C_REGISTER_CALLBACKS 0U /* FMPI2C register callback disabled */
00146 #define USE_HAL_FMPMBUS_REGISTER_CALLBACKS 0U /* FMPMBUS register callback disabled */
00147 #define USE_HAL_I2S_REGISTER_CALLBACKS 0U /* I2S register callback disabled */
00148 #define USE_HAL_IRDA_REGISTER_CALLBACKS 0U /* IRDA register callback disabled */
00149 #define USE_HAL_LPTIM_REGISTER_CALLBACKS 0U /* LPTIM register callback disabled */
00150 #define USE_HAL_LTDC_REGISTER_CALLBACKS 0U /* LTDC register callback disabled */
00151 #define USE_HAL_MMC_REGISTER_CALLBACKS 0U /* MMC register callback disabled */
00152 #define USE_HAL_NAND_REGISTER_CALLBACKS 0U /* NAND register callback disabled */
00153 #define USE_HAL_NOR_REGISTER_CALLBACKS 0U /* NOR register callback disabled */
00154 #define USE_HAL_PCCARD_REGISTER_CALLBACKS 0U /* PCCARD register callback disabled */
00155 #define USE_HAL_PCD_REGISTER_CALLBACKS 0U /* PCD register callback disabled */
00156 #define USE_HAL_QSPI_REGISTER_CALLBACKS 0U /* QSPI register callback disabled */
00157 #define USE_HAL_RNG_REGISTER_CALLBACKS 0U /* RNG register callback disabled */
00158 #define USE_HAL_RTC_REGISTER_CALLBACKS 0U /* RTC register callback disabled */
00159 #define USE_HAL_SAI_REGISTER_CALLBACKS 0U /* SAI register callback disabled */
00160 #define USE_HAL_SD_REGISTER_CALLBACKS 0U /* SD register callback disabled */
00161 #define USE_HAL_SMARTCARD_REGISTER_CALLBACKS 0U /* SMARTCARD register callback disabled */
00162 #define USE_HAL_SDRAM_REGISTER_CALLBACKS 0U /* SDRAM register callback disabled */
00163 #define USE_HAL_SRAM_REGISTER_CALLBACKS 0U /* SRAM register callback disabled */
00164 #define USE_HAL_SPDIFRX_REGISTER_CALLBACKS 0U /* SPDIFRX register callback disabled */
00165 #define USE_HAL_SMBUS_REGISTER_CALLBACKS 0U /* SMBUS register callback disabled */
00166 #define USE_HAL_SPI_REGISTER_CALLBACKS 0U /* SPI register callback disabled */
00167 #define USE_HAL_TIM_REGISTER_CALLBACKS 0U /* TIM register callback disabled */
00168 #define USE_HAL_UART_REGISTER_CALLBACKS 0U /* UART register callback disabled */
00169 #define USE_HAL_USART_REGISTER_CALLBACKS 0U /* USART register callback disabled */
00170 #define USE_HAL_WWDG_REGISTER_CALLBACKS 0U /* WWDG register callback disabled */
00171
00172 /* ##### Assert Selection ##### */
00173 #define USE_FULL_ASSERT 1U
00174
00175 /* ##### Ethernet peripheral configuration ##### */
```

```
00206 /* Section 1 : Ethernet peripheral configuration */
00207
00208 /* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
00209 #define MAC_ADDR0 2U
00210 #define MAC_ADDR1 0U
00211 #define MAC_ADDR2 0U
00212 #define MAC_ADDR3 0U
00213 #define MAC_ADDR4 0U
00214 #define MAC_ADDR5 0U
00215
00216 /* Definition of the Ethernet driver buffers size and count */
00217 #define ETH_RX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for receive */
00218 #define ETH_TX_BUF_SIZE ETH_MAX_PACKET_SIZE /* buffer size for transmit */
00219 #define ETH_RXBUFNB 4U /* 4 Rx buffers of size ETH_RX_BUF_SIZE */
00220 #define ETH_TXBUFNB 4U /* 4 Tx buffers of size ETH_TX_BUF_SIZE */
00221
00222 /* Section 2: PHY configuration section */
00223
00224 /* DP83848_PHY_ADDRESS Address*/
00225 #define DP83848_PHY_ADDRESS
00226 /* PHY Reset delay these values are based on a 1 ms SysTick interrupt*/
00227 #define PHY_RESET_DELAY 0x000000FFU
00228 /* PHY Configuration delay */
00229 #define PHY_CONFIG_DELAY 0x000000FFU
00230
00231 #define PHY_READ_TO 0x0000FFFFU
00232 #define PHY_WRITE_TO 0x0000FFFFU
00233
00234 /* Section 3: Common PHY Registers */
00235
00236 #define PHY_BCR ((uint16_t)0x0000U)
00237 #define PHY_BSR ((uint16_t)0x0001U)
00239 #define PHY_RESET ((uint16_t)0x8000U)
00240 #define PHY_LOOPBACK ((uint16_t)0x4000U)
00241 #define PHY_FULLDUPLEX_100M ((uint16_t)0x2100U)
00242 #define PHY_HALFDUPLEX_100M ((uint16_t)0x2000U)
00243 #define PHY_FULLDUPLEX_10M ((uint16_t)0x0100U)
00244 #define PHY_HALFDUPLEX_10M ((uint16_t)0x0000U)
00245 #define PHY_AUTONEGOTIATION ((uint16_t)0x1000U)
00246 #define PHY_RESTART_AUTONEGOTIATION ((uint16_t)0x0200U)
00247 #define PHY_POWERDOWN ((uint16_t)0x0800U)
00248 #define PHY_ISOLATE ((uint16_t)0x0400U)
00250 #define PHY_AUTONEGO_COMPLETE ((uint16_t)0x0020U)
00251 #define PHY_LINKED_STATUS ((uint16_t)0x0004U)
00252 #define PHY_JABBER_DETECTION ((uint16_t)0x0002U)
00254 /* Section 4: Extended PHY Registers */
00255 #define PHY_SR ((uint16_t))
00257 #define PHY_SPEED_STATUS ((uint16_t))
00258 #define PHY_DUPLEX_STATUS ((uint16_t))
00260 /* ##### SPI peripheral configuration ##### */
00261
00262 /* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
00263 * Activated: CRC code is present inside driver
00264 * Deactivated: CRC code cleaned from driver
00265 */
00266
00267 #define USE_SPI_CRC 0U
00268
00269 /* Includes -----*/
00274 #ifdef HAL_RCC_MODULE_ENABLED
00275 #include "stm32f4xx_hal_rcc.h"
00276 #endif /* HAL_RCC_MODULE_ENABLED */
00277
00278 #ifdef HAL_GPIO_MODULE_ENABLED
00279 #include "stm32f4xx_hal_gpio.h"
00280 #endif /* HAL_GPIO_MODULE_ENABLED */
00281
00282 #ifdef HAL_EXTI_MODULE_ENABLED
00283 #include "stm32f4xx_hal_exti.h"
00284 #endif /* HAL_EXTI_MODULE_ENABLED */
00285
00286 #ifdef HAL_DMA_MODULE_ENABLED
00287 #include "stm32f4xx_hal_dma.h"
00288 #endif /* HAL_DMA_MODULE_ENABLED */
00289
00290 #ifdef HAL_CORTEX_MODULE_ENABLED
00291 #include "stm32f4xx_hal_cortex.h"
00292 #endif /* HAL_CORTEX_MODULE_ENABLED */
00293
00294 #ifdef HAL_ADC_MODULE_ENABLED
00295 #include "stm32f4xx_hal_adc.h"
00296 #endif /* HAL_ADC_MODULE_ENABLED */
00297
00298 #ifdef HAL_CAN_MODULE_ENABLED
00299 #include "stm32f4xx_hal_can.h"
00300 #endif /* HAL_CAN_MODULE_ENABLED */
00301
```

```
00302 #ifdef HAL_CAN_LEGACY_MODULE_ENABLED
00303     #include "stm32f4xx_hal_can_legacy.h"
00304 #endif /* HAL_CAN_LEGACY_MODULE_ENABLED */
00305
00306 #ifdef HAL_CRC_MODULE_ENABLED
00307     #include "stm32f4xx_hal_crc.h"
00308 #endif /* HAL_CRC_MODULE_ENABLED */
00309
00310 #ifdef HAL_Cryp_MODULE_ENABLED
00311     #include "stm32f4xx_hal_cryp.h"
00312 #endif /* HAL_Cryp_MODULE_ENABLED */
00313
00314 #ifdef HAL_DMA2D_MODULE_ENABLED
00315     #include "stm32f4xx_hal_dma2d.h"
00316 #endif /* HAL_DMA2D_MODULE_ENABLED */
00317
00318 #ifdef HAL_DAC_MODULE_ENABLED
00319     #include "stm32f4xx_hal_dac.h"
00320 #endif /* HAL_DAC_MODULE_ENABLED */
00321
00322 #ifdef HAL_DCMI_MODULE_ENABLED
00323     #include "stm32f4xx_hal_dcmi.h"
00324 #endif /* HAL_DCMI_MODULE_ENABLED */
00325
00326 #ifdef HAL_ETH_MODULE_ENABLED
00327     #include "stm32f4xx_hal_eth.h"
00328 #endif /* HAL_ETH_MODULE_ENABLED */
00329
00330 #ifdef HAL_ETH_LEGACY_MODULE_ENABLED
00331     #include "stm32f4xx_hal_eth_legacy.h"
00332 #endif /* HAL_ETH_LEGACY_MODULE_ENABLED */
00333
00334 #ifdef HAL_FLASH_MODULE_ENABLED
00335     #include "stm32f4xx_hal_flash.h"
00336 #endif /* HAL_FLASH_MODULE_ENABLED */
00337
00338 #ifdef HAL_SRAM_MODULE_ENABLED
00339     #include "stm32f4xx_hal_sram.h"
00340 #endif /* HAL_SRAM_MODULE_ENABLED */
00341
00342 #ifdef HAL_NOR_MODULE_ENABLED
00343     #include "stm32f4xx_hal_nor.h"
00344 #endif /* HAL_NOR_MODULE_ENABLED */
00345
00346 #ifdef HAL_NAND_MODULE_ENABLED
00347     #include "stm32f4xx_hal_nand.h"
00348 #endif /* HAL_NAND_MODULE_ENABLED */
00349
00350 #ifdef HAL_PCCARD_MODULE_ENABLED
00351     #include "stm32f4xx_hal_pccard.h"
00352 #endif /* HAL_PCCARD_MODULE_ENABLED */
00353
00354 #ifdef HAL_SDRAM_MODULE_ENABLED
00355     #include "stm32f4xx_hal_sdram.h"
00356 #endif /* HAL_SDRAM_MODULE_ENABLED */
00357
00358 #ifdef HAL_HASH_MODULE_ENABLED
00359     #include "stm32f4xx_hal_hash.h"
00360 #endif /* HAL_HASH_MODULE_ENABLED */
00361
00362 #ifdef HAL_I2C_MODULE_ENABLED
00363     #include "stm32f4xx_hal_i2c.h"
00364 #endif /* HAL_I2C_MODULE_ENABLED */
00365
00366 #ifdef HAL_SMBUS_MODULE_ENABLED
00367     #include "stm32f4xx_hal_smbus.h"
00368 #endif /* HAL_SMBUS_MODULE_ENABLED */
00369
00370 #ifdef HAL_I2S_MODULE_ENABLED
00371     #include "stm32f4xx_hal_i2s.h"
00372 #endif /* HAL_I2S_MODULE_ENABLED */
00373
00374 #ifdef HAL_IWDG_MODULE_ENABLED
00375     #include "stm32f4xx_hal_iwdg.h"
00376 #endif /* HAL_IWDG_MODULE_ENABLED */
00377
00378 #ifdef HAL_LTDC_MODULE_ENABLED
00379     #include "stm32f4xx_hal_ltdc.h"
00380 #endif /* HAL_LTDC_MODULE_ENABLED */
00381
00382 #ifdef HAL_PWR_MODULE_ENABLED
00383     #include "stm32f4xx_hal_pwr.h"
00384 #endif /* HAL_PWR_MODULE_ENABLED */
00385
00386 #ifdef HAL_RNG_MODULE_ENABLED
00387     #include "stm32f4xx_hal_rng.h"
00388 #endif /* HAL_RNG_MODULE_ENABLED */
```

```
00389
00390 #ifndef HAL_RTC_MODULE_ENABLED
00391 #include "stm32f4xx_hal_rtc.h"
00392 #endif /* HAL_RTC_MODULE_ENABLED */
00393
00394 #ifndef HAL_SAI_MODULE_ENABLED
00395 #include "stm32f4xx_hal_sai.h"
00396 #endif /* HAL_SAI_MODULE_ENABLED */
00397
00398 #ifndef HAL_SD_MODULE_ENABLED
00399 #include "stm32f4xx_hal_sd.h"
00400 #endif /* HAL_SD_MODULE_ENABLED */
00401
00402 #ifndef HAL_SPI_MODULE_ENABLED
00403 #include "stm32f4xx_hal_spi.h"
00404 #endif /* HAL_SPI_MODULE_ENABLED */
00405
00406 #ifndef HAL_TIM_MODULE_ENABLED
00407 #include "stm32f4xx_hal_tim.h"
00408 #endif /* HAL_TIM_MODULE_ENABLED */
00409
00410 #ifndef HAL_UART_MODULE_ENABLED
00411 #include "stm32f4xx_hal_uart.h"
00412 #endif /* HAL_UART_MODULE_ENABLED */
00413
00414 #ifndef HAL_USART_MODULE_ENABLED
00415 #include "stm32f4xx_hal_usart.h"
00416 #endif /* HAL_USART_MODULE_ENABLED */
00417
00418 #ifndef HAL_IRDA_MODULE_ENABLED
00419 #include "stm32f4xx_hal_irda.h"
00420 #endif /* HAL_IRDA_MODULE_ENABLED */
00421
00422 #ifndef HAL_SMARTCARD_MODULE_ENABLED
00423 #include "stm32f4xx_hal_smartcard.h"
00424 #endif /* HAL_SMARTCARD_MODULE_ENABLED */
00425
00426 #ifndef HAL_WWDG_MODULE_ENABLED
00427 #include "stm32f4xx_hal_wwdg.h"
00428 #endif /* HAL_WWDG_MODULE_ENABLED */
00429
00430 #ifndef HAL_PCD_MODULE_ENABLED
00431 #include "stm32f4xx_hal_pcd.h"
00432 #endif /* HAL_PCD_MODULE_ENABLED */
00433
00434 #ifndef HAL_HCD_MODULE_ENABLED
00435 #include "stm32f4xx_hal_hcd.h"
00436 #endif /* HAL_HCD_MODULE_ENABLED */
00437
00438 #ifndef HAL_DSI_MODULE_ENABLED
00439 #include "stm32f4xx_hal_dsi.h"
00440 #endif /* HAL_DSI_MODULE_ENABLED */
00441
00442 #ifndef HAL_QSPI_MODULE_ENABLED
00443 #include "stm32f4xx_hal_qspi.h"
00444 #endif /* HAL_QSPI_MODULE_ENABLED */
00445
00446 #ifndef HAL_CEC_MODULE_ENABLED
00447 #include "stm32f4xx_hal_cec.h"
00448 #endif /* HAL_CEC_MODULE_ENABLED */
00449
00450 #ifndef HAL_FMPI2C_MODULE_ENABLED
00451 #include "stm32f4xx_hal_fmpi2c.h"
00452 #endif /* HAL_FMPI2C_MODULE_ENABLED */
00453
00454 #ifndef HAL_FMPUSBUS_MODULE_ENABLED
00455 #include "stm32f4xx_hal_fmpsmbus.h"
00456 #endif /* HAL_FMPUSBUS_MODULE_ENABLED */
00457
00458 #ifndef HAL_SPDIFRX_MODULE_ENABLED
00459 #include "stm32f4xx_hal_spdifrx.h"
00460 #endif /* HAL_SPDIFRX_MODULE_ENABLED */
00461
00462 #ifndef HAL_DFSDM_MODULE_ENABLED
00463 #include "stm32f4xx_hal_dfstdm.h"
00464 #endif /* HAL_DFSDM_MODULE_ENABLED */
00465
00466 #ifndef HAL_LPTIM_MODULE_ENABLED
00467 #include "stm32f4xx_hal_lptim.h"
00468 #endif /* HAL_LPTIM_MODULE_ENABLED */
00469
00470 #ifndef HAL_MMC_MODULE_ENABLED
00471 #include "stm32f4xx_hal_mmc.h"
00472 #endif /* HAL_MMC_MODULE_ENABLED */
00473
00474 /* Exported macro -----*/
00475 #ifndef USE_FULL_ASSERT
```

```

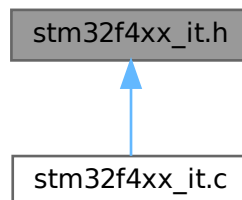
00484 #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__, __LINE__))
00485 /* Exported functions ----- */
00486 void assert_failed(uint8_t* file, uint32_t line);
00487 #else
00488 #define assert_param(expr) ((void)0U)
00489 #endif /* USE_FULL_ASSERT */
00490
00491 #ifdef __cplusplus
00492 }
00493 #endif
00494
00495 #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

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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.](#)

```
00001 /* USER CODE BEGIN Header */
00018 /* USER CODE END Header */
00019
00020 /* Define to prevent recursive inclusion -----*/
00021 #ifndef __STM32F4xx_IT_H
00022 #define __STM32F4xx_IT_H
00023
00024 #ifdef __cplusplus
00025     extern "C" {
00026 #endif
00027
00028 /* Private includes -----*/
00029 /* USER CODE BEGIN Includes */
00030
00031 /* USER CODE END Includes */
00032
00033 /* Exported types -----*/
00034 /* USER CODE BEGIN ET */
00035
00036 /* USER CODE END ET */
00037
00038 /* Exported constants -----*/
00039 /* USER CODE BEGIN EC */
00040
00041 /* USER CODE END EC */
00042
00043 /* Exported macro -----*/
00044 /* USER CODE BEGIN EM */
00045
```

```

00046 /* USER CODE END EM */
00047
00048 /* Exported functions prototypes -----*/
00049 void NMI_Handler(void);
00050 void HardFault_Handler(void);
00051 void MemManage_Handler(void);
00052 void BusFault_Handler(void);
00053 void UsageFault_Handler(void);
00054 void SVC_Handler(void);
00055 void DebugMon_Handler(void);
00056 void PendSV_Handler(void);
00057 void SysTick_Handler(void);
00058 /* USER CODE BEGIN EFP */
00059
00060 /* USER CODE END EFP */
00061
00062 #ifdef __cplusplus
00063 }
00064 #endif
00065
00066 #endif /* __STM32F4xx_IT_H */

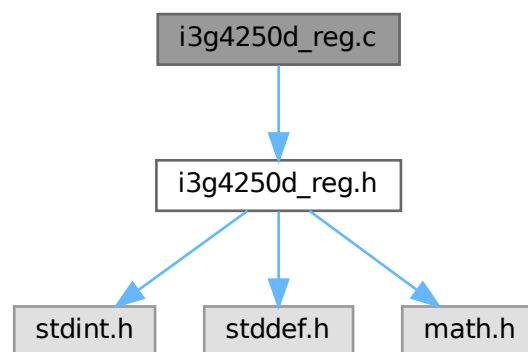
```

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 aml.[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)
- Overflow 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

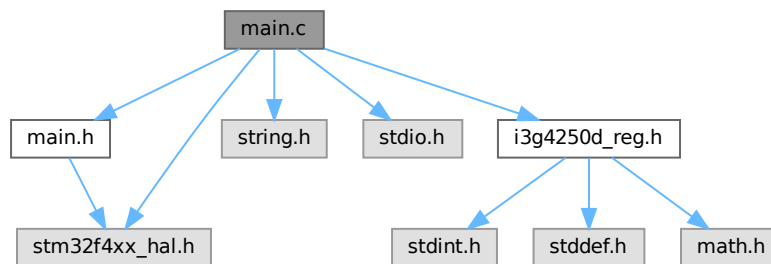
© Copyright (c) 2021 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

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 [hspl1](#)
- UART_HandleTypeDef [huart2](#)

6.10.1 Detailed Description

: Main program body

Attention

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

<i>int</i>	
------------	--

System Clock Configuration

Return values

<i>None</i>	
-------------	--

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

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

USART2 Initialization Function

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

GPIO Initialization Function

Parameters

<i>None</i>	
-------------	--

Return values

<i>None</i>	
-------------	--

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	<i>handle</i>	SPI handle to use for communication.
in	<i>reg</i>	Register address to which data needs to be written.
in	<i>bufp</i>	Pointer to the data buffer to be written.
in	<i>len</i>	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	<i>handle</i>	SPI handle to use for communication.
in	<i>reg</i>	Register address from which data needs to be read.
out	<i>bufp</i>	Pointer to the buffer where the read data will be stored.
in	<i>len</i>	Length of the data buffer in bytes to be read.

Returns

int32_t 0 on success, error code otherwise.

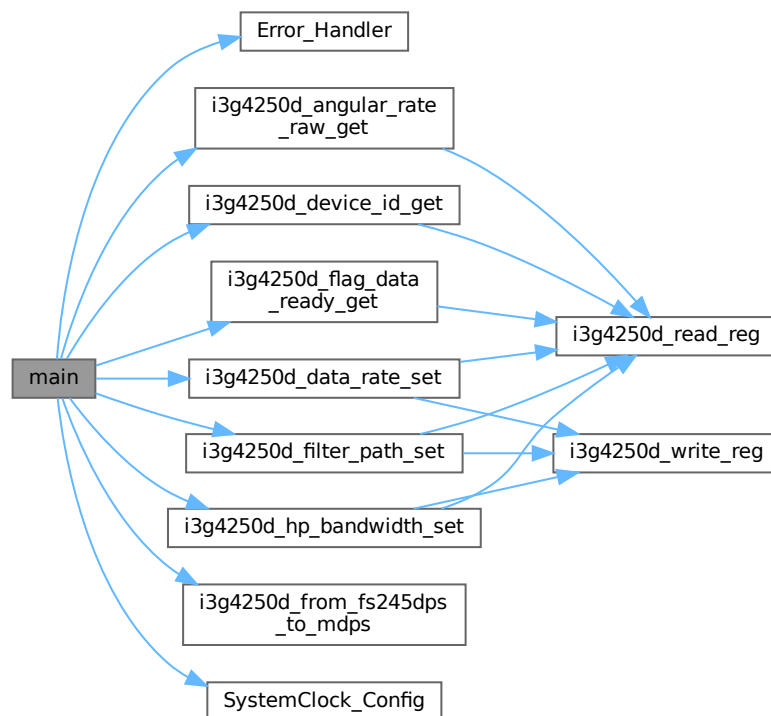
This function is executed in case of error occurrence.

Return values

<i>None</i>	
-------------	--

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:

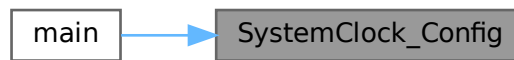


6.10.3.2 SystemClock_Config()

```
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
```

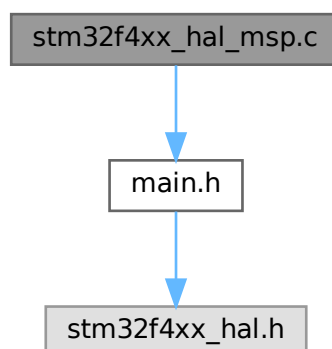
Referenced by [main\(\)](#).

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)
SPI MSP De-Initialization This function freeze the hardware resources used in this example.
- void [HAL_SPI_MspDeInit](#) (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

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

<i>hspi</i>	SPI handle pointer
-------------	--------------------

Return values

<i>None</i>	
-------------	--

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

<i>hspi</i>	SPI handle pointer
-------------	--------------------

Return values

<i>None</i>	
-------------	--

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

<i>huart</i>	UART handle pointer
--------------	---------------------

Return values

<i>None</i>	
-------------	--

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

<i>huart</i>	UART handle pointer
--------------	---------------------

Return values

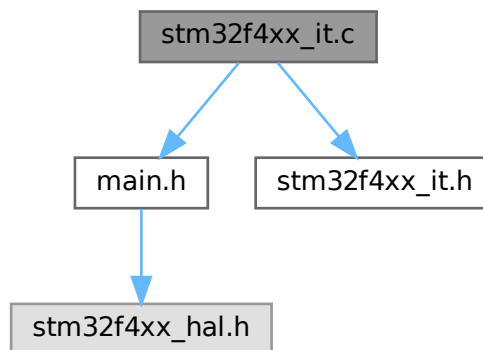
None	
------	--

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

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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 )
```

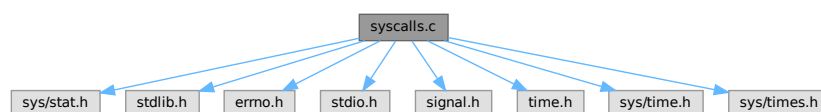
This function handles Undefined instruction or illegal state.

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

Copyright (c) 2020-2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

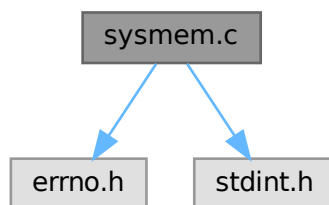
6.13.3.1 environ

```
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

Copyright (c) 2024 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

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

<i>incr</i>	Memory size
-------------	-------------

Returns

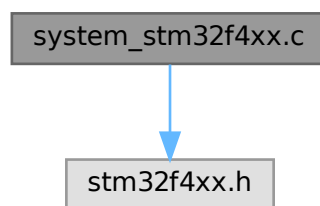
Pointer to allocated memory

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

Copyright (c) 2017 STMicroelectronics. All rights reserved.

This software is licensed under terms that can be found in the LICENSE file in the root directory of this software component. If no LICENSE file comes with this software, it is provided AS-IS.

Index

- `__attribute__`
 - `syscalls.c`, [218](#)
 - `__io_getchar`
 - `syscalls.c`, [218](#)
 - `__io_putchar`
 - `syscalls.c`, [218](#)
 - `__weak`
 - `I3G4250D`, [14](#)
 - `_close`
 - `syscalls.c`, [218](#)
 - `_execve`
 - `syscalls.c`, [218](#)
 - `_exit`
 - `syscalls.c`, [219](#)
 - `_fork`
 - `syscalls.c`, [219](#)
 - `_fstat`
 - `syscalls.c`, [219](#)
 - `_getpid`
 - `syscalls.c`, [219](#)
 - `_isatty`
 - `syscalls.c`, [219](#)
 - `_kill`
 - `syscalls.c`, [219](#)
 - `_link`
 - `syscalls.c`, [220](#)
 - `_lseek`
 - `syscalls.c`, [220](#)
 - `_open`
 - `syscalls.c`, [220](#)
 - `_sbrk`
 - `sysmem.c`, [222](#)
 - `_stat`
 - `syscalls.c`, [220](#)
 - `_times`
 - `syscalls.c`, [220](#)
 - `_unlink`
 - `syscalls.c`, [220](#)
 - `_wait`
 - `syscalls.c`, [221](#)
 - `_xor`
 - `i3g4250d_status_reg_t`, [161](#)
- `address`
 - `ucf_line_t`, [163](#)
- `address-data structure definition`, [130](#)
- `AHBPrescTable`
 - `STM32F4xx_System_Private_Variables`, [136](#)
- `and_or`
 - `i3g4250d_int1_cfg_t`, [149](#)
- `APBPrescTable`
 - `STM32F4xx_System_Private_Variables`, [136](#)
- `assert_param`
 - `stm32f4xx_hal_conf.h`, [183](#)
- `bit0`
 - `bitwise_t`, [141](#)
- `bit1`
 - `bitwise_t`, [141](#)
- `bit2`
 - `bitwise_t`, [141](#)
- `bit3`
 - `bitwise_t`, [141](#)
- `bit4`
 - `bitwise_t`, [142](#)
- `bit5`
 - `bitwise_t`, [142](#)
- `bit6`
 - `bitwise_t`, [142](#)
- `bit7`
 - `bitwise_t`, [142](#)
- `bitwise`
 - `i3g4250d_reg_t`, [159](#)
- `bitwise_t`, [141](#)
 - `bit0`, [141](#)
 - `bit1`, [141](#)
 - `bit2`, [141](#)
 - `bit3`, [141](#)
 - `bit4`, [142](#)
 - `bit5`, [142](#)
 - `bit6`, [142](#)
 - `bit7`, [142](#)
- `ble`
 - `i3g4250d_ctrl_reg4_t`, [145](#)
- `boot`
 - `i3g4250d_ctrl_reg5_t`, [147](#)
- `BOOT_TIME`
 - `main.c`, [208](#)
- `BusFault_Handler`
 - `stm32f4xx_it.c`, [215](#)
 - `stm32f4xx_it.h`, [201](#)
- `bw`
 - `i3g4250d_ctrl_reg1_t`, [142](#)
- `byte`
 - `i3g4250d_reg_t`, [159](#)
- `CMSIS`, [133](#)
- `ctrl_reg1`
 - `i3g4250d_reg_t`, [159](#)
- `ctrl_reg2`

- i3g4250d_reg_t, [159](#)
- ctrl_reg3
 - i3g4250d_reg_t, [159](#)
- ctrl_reg4
 - i3g4250d_reg_t, [159](#)
- ctrl_reg5
 - i3g4250d_reg_t, [159](#)
- d
 - i3g4250d_int1_duration_t, [150](#)
- data
 - ucf_line_t, [163](#)
- DATA_CACHE_ENABLE
 - stm32f4xx_hal_conf.h, [183](#)
- DebugMon_Handler
 - stm32f4xx_it.c, [215](#)
 - stm32f4xx_it.h, [201](#)
- definitions, [128](#)
 - DRV_BIG_ENDIAN, [128](#)
 - DRV_BYTE_ORDER, [128](#)
 - DRV_LITTLE_ENDIAN, [129](#)
- DP83848_PHY_ADDRESS
 - stm32f4xx_hal_conf.h, [184](#)
- dr
 - i3g4250d_ctrl_reg1_t, [142](#)
- DRV_BIG_ENDIAN
 - definitions, [128](#)
- DRV_BYTE_ORDER
 - definitions, [128](#)
- DRV_LITTLE_ENDIAN
 - definitions, [129](#)
- empty
 - i3g4250d_fifo_src_reg_t, [148](#)
- environ
 - syscalls.c, [221](#)
- Error_Handler
 - main.h, [180](#)
- ETH_RX_BUF_SIZE
 - stm32f4xx_hal_conf.h, [184](#)
- ETH_RXBUFNB
 - stm32f4xx_hal_conf.h, [184](#)
- ETH_TX_BUF_SIZE
 - stm32f4xx_hal_conf.h, [184](#)
- ETH_TXBUFNB
 - stm32f4xx_hal_conf.h, [184](#)
- EXTERNAL_CLOCK_VALUE
 - stm32f4xx_hal_conf.h, [184](#)
- fifo_ctrl_reg
 - i3g4250d_reg_t, [159](#)
- fifo_en
 - i3g4250d_ctrl_reg5_t, [147](#)
- fifo_src_reg
 - i3g4250d_reg_t, [160](#)
- fm
 - i3g4250d_fifo_ctrl_reg_t, [148](#)
- fs
 - i3g4250d_ctrl_reg4_t, [145](#)
- fss
 - i3g4250d_fifo_src_reg_t, [148](#)
- GYRO_CS_OUT_GPIO_Port
 - main.h, [180](#)
- GYRO_CS_OUT_Pin
 - main.h, [180](#)
- h_lactive
 - i3g4250d_ctrl_reg3_t, [144](#)
- HAL_CORTEX_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [184](#)
- HAL_DMA_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [184](#)
- HAL_EXTI_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [184](#)
- HAL_FLASH_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_GPIO_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_MspInit
 - stm32f4xx_hal_msp.c, [212](#)
- HAL_PWR_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_RCC_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_SPI_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_SPI_MspDeInit
 - stm32f4xx_hal_msp.c, [212](#)
- HAL_SPI_MspInit
 - stm32f4xx_hal_msp.c, [213](#)
- HAL_UART_MODULE_ENABLED
 - stm32f4xx_hal_conf.h, [185](#)
- HAL_UART_MspDeInit
 - stm32f4xx_hal_msp.c, [213](#)
- HAL_UART_MspInit
 - stm32f4xx_hal_msp.c, [213](#)
- handle
 - stmdev_ctx_t, [162](#)
- HardFault_Handler
 - stm32f4xx_it.c, [215](#)
 - stm32f4xx_it.h, [201](#)
- hpcf
 - i3g4250d_ctrl_reg2_t, [143](#)
- hpen
 - i3g4250d_ctrl_reg5_t, [147](#)
- hpm
 - i3g4250d_ctrl_reg2_t, [143](#)
- HSE_STARTUP_TIMEOUT
 - stm32f4xx_hal_conf.h, [185](#)
- HSE_VALUE
 - stm32f4xx_hal_conf.h, [185](#)
 - STM32F4xx_System_Private_Includes, [135](#)
- HSI_VALUE
 - stm32f4xx_hal_conf.h, [186](#)
 - STM32F4xx_System_Private_Includes, [135](#)

- hspi1
 - main.c, [211](#)
- huart2
 - main.c, [211](#)
- i1_boot
 - i3g4250d_ctrl_reg3_t, [144](#)
 - i3g4250d_int1_route_t, [151](#)
- i1_int1
 - i3g4250d_ctrl_reg3_t, [144](#)
 - i3g4250d_int1_route_t, [151](#)
- i2_drdy
 - i3g4250d_ctrl_reg3_t, [144](#)
 - i3g4250d_int2_route_t, [156](#)
- i2_empty
 - i3g4250d_ctrl_reg3_t, [144](#)
 - i3g4250d_int2_route_t, [156](#)
- i2_orun
 - i3g4250d_ctrl_reg3_t, [144](#)
 - i3g4250d_int2_route_t, [156](#)
- i2_wtm
 - i3g4250d_ctrl_reg3_t, [145](#)
 - i3g4250d_int2_route_t, [156](#)
- I3G4250D, [7](#)
 - __weak, [14](#)
 - I3G4250D_2000dps, [18](#)
 - I3G4250D_245dps, [18](#)
 - I3G4250D_500dps, [18](#)
 - I3G4250D_ACTIVE_HIGH, [18](#)
 - I3G4250D_ACTIVE_LOW, [18](#)
 - i3g4250d_and_or_t, [16](#)
 - i3g4250d_angular_rate_raw_get, [20](#)
 - I3G4250D_AUX_LSB_AT_LOW_ADD, [17](#)
 - I3G4250D_AUX_MSB_AT_LOW_ADD, [17](#)
 - i3g4250d_axis_x_data_get, [21](#)
 - i3g4250d_axis_x_data_set, [21](#)
 - i3g4250d_axis_y_data_get, [21](#)
 - i3g4250d_axis_y_data_set, [21](#)
 - i3g4250d_axis_z_data_get, [21](#)
 - i3g4250d_axis_z_data_set, [22](#)
 - i3g4250d_ble_t, [16](#)
 - i3g4250d_boot_get, [22](#)
 - i3g4250d_boot_set, [22](#)
 - i3g4250d_bw_t, [17](#)
 - I3G4250D_CTRL_REG1, [14](#)
 - I3G4250D_CTRL_REG2, [14](#)
 - I3G4250D_CTRL_REG3, [14](#)
 - I3G4250D_CTRL_REG4, [14](#)
 - I3G4250D_CTRL_REG5, [14](#)
 - I3G4250D_CUT_OFF_HIGH, [17](#)
 - I3G4250D_CUT_OFF_LOW, [17](#)
 - I3G4250D_CUT_OFF_MEDIUM, [17](#)
 - I3G4250D_CUT_OFF_VERY_HIGH, [17](#)
 - i3g4250d_data_format_get, [23](#)
 - i3g4250d_data_format_set, [24](#)
 - i3g4250d_data_rate_get, [24](#)
 - i3g4250d_data_rate_set, [25](#)
 - i3g4250d_device_id_get, [26](#)
 - i3g4250d_dr_t, [17](#)
 - I3G4250D_FIFO_BYPASS_MODE, [17](#)
 - I3G4250D_FIFO_CTRL_REG, [14](#)
 - i3g4250d_fifo_data_level_get, [27](#)
 - i3g4250d_fifo_empty_flag_get, [28](#)
 - i3g4250d_fifo_enable_get, [28](#)
 - i3g4250d_fifo_enable_set, [29](#)
 - I3G4250D_FIFO_MODE, [17](#)
 - i3g4250d_fifo_mode_get, [30](#)
 - i3g4250d_fifo_mode_set, [30](#)
 - i3g4250d_fifo_mode_t, [17](#)
 - i3g4250d_fifo_ovr_flag_get, [32](#)
 - I3G4250D_FIFO_SRC_REG, [14](#)
 - I3G4250D_FIFO_STREAM_MODE, [17](#)
 - i3g4250d_fifo_watermark_get, [33](#)
 - i3g4250d_fifo_watermark_set, [33](#)
 - i3g4250d_fifo_wtm_flag_get, [34](#)
 - i3g4250d_filter_path_get, [35](#)
 - i3g4250d_filter_path_internal_get, [35](#)
 - i3g4250d_filter_path_internal_set, [36](#)
 - i3g4250d_filter_path_set, [37](#)
 - i3g4250d_flag_data_ready_get, [38](#)
 - i3g4250d_from_fs245dps_to_mdps, [39](#)
 - i3g4250d_from_lsb_to_celsius, [39](#)
 - i3g4250d_fs_t, [17](#)
 - i3g4250d_full_scale_get, [39](#)
 - i3g4250d_full_scale_set, [40](#)
 - I3G4250D_GY_ST_DISABLE, [20](#)
 - I3G4250D_GY_ST_NEGATIVE, [20](#)
 - I3G4250D_GY_ST_POSITIVE, [20](#)
 - i3g4250d_h_lactive_t, [18](#)
 - I3G4250D_HP_AUTO_RESET_ON_INT, [19](#)
 - i3g4250d_hp_bandwidth_get, [40](#)
 - i3g4250d_hp_bandwidth_set, [41](#)
 - I3G4250D_HP_LEVEL_0, [18](#)
 - I3G4250D_HP_LEVEL_1, [18](#)
 - I3G4250D_HP_LEVEL_2, [18](#)
 - I3G4250D_HP_LEVEL_3, [18](#)
 - I3G4250D_HP_LEVEL_4, [18](#)
 - I3G4250D_HP_LEVEL_5, [18](#)
 - I3G4250D_HP_LEVEL_6, [18](#)
 - I3G4250D_HP_LEVEL_7, [18](#)
 - I3G4250D_HP_LEVEL_8, [18](#)
 - I3G4250D_HP_LEVEL_9, [18](#)
 - i3g4250d_hp_mode_get, [42](#)
 - i3g4250d_hp_mode_set, [43](#)
 - I3G4250D_HP_NORMAL_MODE, [19](#)
 - I3G4250D_HP_NORMAL_MODE_WITH_RST, [19](#)
 - I3G4250D_HP_REFERENCE_SIGNAL, [19](#)
 - i3g4250d_hp_reference_value_get, [43](#)
 - i3g4250d_hp_reference_value_set, [45](#)
 - i3g4250d_hpcf_t, [18](#)
 - i3g4250d_hpm_t, [18](#)
 - I3G4250D_INT1_CFG, [14](#)
 - I3G4250D_INT1_DURATION, [14](#)
 - I3G4250D_INT1_ON_TH_AND, [16](#)
 - I3G4250D_INT1_ON_TH_OR, [16](#)
 - i3g4250d_int1_sel_t, [19](#)
 - I3G4250D_INT1_SRC, [14](#)

I3G4250D_INT1_TSH_XH, 15
 I3G4250D_INT1_TSH_XL, 15
 I3G4250D_INT1_TSH_YH, 15
 I3G4250D_INT1_TSH_YL, 15
 I3G4250D_INT1_TSH_ZH, 15
 I3G4250D_INT1_TSH_ZL, 15
 I3G4250D_INT_LATCHED, 19
 i3g4250d_int_notification_get, 46
 i3g4250d_int_notification_set, 46
 i3g4250d_int_on_threshold_conf_get, 48
 i3g4250d_int_on_threshold_conf_set, 49
 i3g4250d_int_on_threshold_dur_get, 49
 i3g4250d_int_on_threshold_dur_set, 50
 i3g4250d_int_on_threshold_mode_get, 51
 i3g4250d_int_on_threshold_mode_set, 51
 i3g4250d_int_on_threshold_src_get, 52
 I3G4250D_INT_PULSED, 19
 i3g4250d_int_x_threshold_get, 53
 i3g4250d_int_x_threshold_set, 53
 i3g4250d_int_y_threshold_get, 54
 i3g4250d_int_y_threshold_set, 55
 i3g4250d_int_z_threshold_get, 55
 i3g4250d_int_z_threshold_set, 56
 i3g4250d_lir_t, 19
 i3g4250d_lp_bandwidth_get, 57
 i3g4250d_lp_bandwidth_set, 57
 I3G4250D_LPF1_HP_LPF2_ON_INT, 19
 I3G4250D_LPF1_HP_LPF2_ON_OUT, 19
 I3G4250D_LPF1_HP_ON_INT, 19
 I3G4250D_LPF1_HP_ON_OUT, 19
 I3G4250D_LPF1_LPF2_ON_INT, 19
 I3G4250D_LPF1_LPF2_ON_OUT, 19
 I3G4250D_ODR_100Hz, 17
 I3G4250D_ODR_200Hz, 17
 I3G4250D_ODR_400Hz, 17
 I3G4250D_ODR_800Hz, 17
 I3G4250D_ODR_OFF, 17
 I3G4250D_ODR_SLEEP, 17
 I3G4250D_ONLY_LPF1_ON_INT, 19
 I3G4250D_ONLY_LPF1_ON_OUT, 19
 I3G4250D_OPEN_DRAIN, 20
 i3g4250d_out_sel_t, 19
 I3G4250D_OUT_TEMP, 15
 I3G4250D_OUT_X_H, 15
 I3G4250D_OUT_X_L, 15
 I3G4250D_OUT_Y_H, 15
 I3G4250D_OUT_Y_L, 16
 I3G4250D_OUT_Z_H, 16
 I3G4250D_OUT_Z_L, 16
 i3g4250d_pin_int1_route_get, 58
 i3g4250d_pin_int1_route_set, 59
 i3g4250d_pin_int2_route_get, 59
 i3g4250d_pin_int2_route_set, 60
 i3g4250d_pin_mode_get, 61
 i3g4250d_pin_mode_set, 61
 i3g4250d_pin_polarity_get, 63
 i3g4250d_pin_polarity_set, 64
 i3g4250d_pp_od_t, 19
 I3G4250D_PUSH_PULL, 20
 i3g4250d_read_reg, 64
 I3G4250D_REFERENCE, 16
 i3g4250d_self_test_get, 65
 i3g4250d_self_test_set, 66
 i3g4250d_sim_t, 20
 I3G4250D_SPI_3_WIRE, 20
 I3G4250D_SPI_4_WIRE, 20
 i3g4250d_spi_mode_get, 66
 i3g4250d_spi_mode_set, 67
 i3g4250d_st_t, 20
 I3G4250D_STATUS_REG, 16
 i3g4250d_status_reg_get, 68
 i3g4250d_temperature_raw_get, 68
 I3G4250D_WHO_AM_I, 16
 i3g4250d_write_reg, 69
 I3G4250D_2000dps
 I3G4250D, 18
 I3G4250D_245dps
 I3G4250D, 18
 I3G4250D_500dps
 I3G4250D, 18
 I3G4250D_ACTIVE_HIGH
 I3G4250D, 18
 I3G4250D_ACTIVE_LOW
 I3G4250D, 18
 i3g4250d_and_or_t
 I3G4250D, 16
 i3g4250d_angular_rate_raw_get
 I3G4250D, 20
 I3G4250D_Dataoutput, 82
 I3G4250D_AUX_LSB_AT_LOW_ADD
 I3G4250D, 17
 I3G4250D_AUX_MSB_AT_LOW_ADD
 I3G4250D, 17
 i3g4250d_axis_x_data_get
 I3G4250D, 21
 i3g4250d_axis_x_data_set
 I3G4250D, 21
 i3g4250d_axis_y_data_get
 I3G4250D, 21
 i3g4250d_axis_y_data_set
 I3G4250D, 21
 i3g4250d_axis_z_data_get
 I3G4250D, 21
 i3g4250d_axis_z_data_set
 I3G4250D, 22
 i3g4250d_ble_t
 I3G4250D, 16
 i3g4250d_boot_get
 I3G4250D, 22
 I3G4250D_common, 84
 i3g4250d_boot_set
 I3G4250D, 22
 I3G4250D_common, 85
 i3g4250d_bw_t
 I3G4250D, 17
 I3G4250D_common, 83

- i3g4250d_boot_get, [84](#)
- i3g4250d_boot_set, [85](#)
- i3g4250d_data_format_get, [85](#)
- i3g4250d_data_format_set, [87](#)
- i3g4250d_device_id_get, [88](#)
- i3g4250d_self_test_get, [89](#)
- i3g4250d_self_test_set, [89](#)
- I3G4250D_CTRL_REG1
 - I3G4250D, [14](#)
- i3g4250d_ctrl_reg1_t, [142](#)
 - bw, [142](#)
 - dr, [142](#)
 - pd, [143](#)
- I3G4250D_CTRL_REG2
 - I3G4250D, [14](#)
- i3g4250d_ctrl_reg2_t, [143](#)
 - hpcf, [143](#)
 - hpm, [143](#)
 - not_used_01, [143](#)
- I3G4250D_CTRL_REG3
 - I3G4250D, [14](#)
- i3g4250d_ctrl_reg3_t, [144](#)
 - h_lactive, [144](#)
 - i1_boot, [144](#)
 - i1_int1, [144](#)
 - i2_drdy, [144](#)
 - i2_empty, [144](#)
 - i2_orun, [144](#)
 - i2_wtm, [145](#)
 - pp_od, [145](#)
- I3G4250D_CTRL_REG4
 - I3G4250D, [14](#)
- i3g4250d_ctrl_reg4_t, [145](#)
 - ble, [145](#)
 - fs, [145](#)
 - not_used_01, [146](#)
 - not_used_02, [146](#)
 - sim, [146](#)
 - st, [146](#)
- I3G4250D_CTRL_REG5
 - I3G4250D, [14](#)
- i3g4250d_ctrl_reg5_t, [146](#)
 - boot, [147](#)
 - fifo_en, [147](#)
 - hpen, [147](#)
 - int1_sel, [147](#)
 - not_used_01, [147](#)
 - out_sel, [147](#)
- I3G4250D_CUT_OFF_HIGH
 - I3G4250D, [17](#)
- I3G4250D_CUT_OFF_LOW
 - I3G4250D, [17](#)
- I3G4250D_CUT_OFF_MEDIUM
 - I3G4250D, [17](#)
- I3G4250D_CUT_OFF_VERY_HIGH
 - I3G4250D, [17](#)
- i3g4250d_data_format_get
 - I3G4250D, [23](#)
- I3G4250D_common, [85](#)
- i3g4250d_data_format_set
 - I3G4250D, [24](#)
 - I3G4250D_common, [87](#)
- I3G4250D_data_generation, [75](#)
 - i3g4250d_data_rate_get, [75](#)
 - i3g4250d_data_rate_set, [76](#)
 - i3g4250d_flag_data_ready_get, [77](#)
 - i3g4250d_full_scale_get, [78](#)
 - i3g4250d_full_scale_set, [79](#)
 - i3g4250d_status_reg_get, [79](#)
- i3g4250d_data_rate_get
 - I3G4250D, [24](#)
 - I3G4250D_data_generation, [75](#)
- i3g4250d_data_rate_set
 - I3G4250D, [25](#)
 - I3G4250D_data_generation, [76](#)
- I3G4250D_Dataoutput, [81](#)
 - i3g4250d_angular_rate_raw_get, [82](#)
 - i3g4250d_temperature_raw_get, [82](#)
- i3g4250d_device_id_get
 - I3G4250D, [26](#)
 - I3G4250D_common, [88](#)
- i3g4250d_dr_t
 - I3G4250D, [17](#)
- I3G4250D_fifo, [121](#)
 - i3g4250d_fifo_data_level_get, [122](#)
 - i3g4250d_fifo_empty_flag_get, [122](#)
 - i3g4250d_fifo_enable_get, [123](#)
 - i3g4250d_fifo_enable_set, [123](#)
 - i3g4250d_fifo_mode_get, [124](#)
 - i3g4250d_fifo_mode_set, [125](#)
 - i3g4250d_fifo_ovr_flag_get, [125](#)
 - i3g4250d_fifo_watermark_get, [126](#)
 - i3g4250d_fifo_watermark_set, [127](#)
 - i3g4250d_fifo_wtm_flag_get, [127](#)
- I3G4250D_FIFO_BYPASS_MODE
 - I3G4250D, [17](#)
- I3G4250D_FIFO_CTRL_REG
 - I3G4250D, [14](#)
- i3g4250d_fifo_ctrl_reg_t, [147](#)
 - fm, [148](#)
 - wtm, [148](#)
- i3g4250d_fifo_data_level_get
 - I3G4250D, [27](#)
 - I3G4250D_fifo, [122](#)
- i3g4250d_fifo_empty_flag_get
 - I3G4250D, [28](#)
 - I3G4250D_fifo, [122](#)
- i3g4250d_fifo_enable_get
 - I3G4250D, [28](#)
 - I3G4250D_fifo, [123](#)
- i3g4250d_fifo_enable_set
 - I3G4250D, [29](#)
 - I3G4250D_fifo, [123](#)
- I3G4250D_FIFO_MODE
 - I3G4250D, [17](#)
- i3g4250d_fifo_mode_get

- I3G4250D, 30
- I3G4250D_fifo, 124
- i3g4250d_fifo_mode_set
 - I3G4250D, 30
 - I3G4250D_fifo, 125
- i3g4250d_fifo_mode_t
 - I3G4250D, 17
- i3g4250d_fifo_ovr_flag_get
 - I3G4250D, 32
 - I3G4250D_fifo, 125
- I3G4250D_FIFO_SRC_REG
 - I3G4250D, 14
- i3g4250d_fifo_src_reg_t, 148
 - empty, 148
 - fss, 148
 - ovrn, 148
 - wtm, 149
- I3G4250D_FIFO_STREAM_MODE
 - I3G4250D, 17
- i3g4250d_fifo_watermark_get
 - I3G4250D, 33
 - I3G4250D_fifo, 126
- i3g4250d_fifo_watermark_set
 - I3G4250D, 33
 - I3G4250D_fifo, 127
- i3g4250d_fifo_wtm_flag_get
 - I3G4250D, 34
 - I3G4250D_fifo, 127
- i3g4250d_filter_path_get
 - I3G4250D, 35
 - I3G4250D_filters, 91
- i3g4250d_filter_path_internal_get
 - I3G4250D, 35
 - I3G4250D_filters, 92
- i3g4250d_filter_path_internal_set
 - I3G4250D, 36
 - I3G4250D_filters, 92
- i3g4250d_filter_path_set
 - I3G4250D, 37
 - I3G4250D_filters, 94
- I3G4250D_filters, 90
 - i3g4250d_filter_path_get, 91
 - i3g4250d_filter_path_internal_get, 92
 - i3g4250d_filter_path_internal_set, 92
 - i3g4250d_filter_path_set, 94
 - i3g4250d_hp_bandwidth_get, 95
 - i3g4250d_hp_bandwidth_set, 96
 - i3g4250d_hp_mode_get, 97
 - i3g4250d_hp_mode_set, 97
 - i3g4250d_hp_reference_value_get, 98
 - i3g4250d_hp_reference_value_set, 99
 - i3g4250d_lp_bandwidth_get, 99
 - i3g4250d_lp_bandwidth_set, 100
- i3g4250d_flag_data_ready_get
 - I3G4250D, 38
 - I3G4250D_data_generation, 77
- i3g4250d_from_fs245dps_to_mdps
 - I3G4250D, 39
- I3G4250D_Sensitivity, 74
- i3g4250d_from_lsb_to_celsius
 - I3G4250D, 39
 - I3G4250D_Sensitivity, 74
- i3g4250d_fs_t
 - I3G4250D, 17
- i3g4250d_full_scale_get
 - I3G4250D, 39
 - I3G4250D_data_generation, 78
- i3g4250d_full_scale_set
 - I3G4250D, 40
 - I3G4250D_data_generation, 79
- I3G4250D_GY_ST_DISABLE
 - I3G4250D, 20
- I3G4250D_GY_ST_NEGATIVE
 - I3G4250D, 20
- I3G4250D_GY_ST_POSITIVE
 - I3G4250D, 20
- i3g4250d_h_lactive_t
 - I3G4250D, 18
- I3G4250D_HP_AUTO_RESET_ON_INT
 - I3G4250D, 19
- i3g4250d_hp_bandwidth_get
 - I3G4250D, 40
 - I3G4250D_filters, 95
- i3g4250d_hp_bandwidth_set
 - I3G4250D, 41
 - I3G4250D_filters, 96
- I3G4250D_HP_LEVEL_0
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_1
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_2
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_3
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_4
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_5
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_6
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_7
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_8
 - I3G4250D, 18
- I3G4250D_HP_LEVEL_9
 - I3G4250D, 18
- i3g4250d_hp_mode_get
 - I3G4250D, 42
 - I3G4250D_filters, 97
- i3g4250d_hp_mode_set
 - I3G4250D, 43
 - I3G4250D_filters, 97
- I3G4250D_HP_NORMAL_MODE
 - I3G4250D, 19
- I3G4250D_HP_NORMAL_MODE_WITH_RST
 - I3G4250D, 19

- I3G4250D_HP_REFERENCE_SIGNAL
 - I3G4250D, [19](#)
- i3g4250d_hp_reference_value_get
 - I3G4250D, [43](#)
 - I3G4250D_filters, [98](#)
- i3g4250d_hp_reference_value_set
 - I3G4250D, [45](#)
 - I3G4250D_filters, [99](#)
- i3g4250d_hpcf_t
 - I3G4250D, [18](#)
- i3g4250d_hpm_t
 - I3G4250D, [18](#)
- I3G4250D_I2C_ADD_H
 - I3g4250d_Infos, [132](#)
- I3G4250D_I2C_ADD_L
 - I3g4250d_Infos, [132](#)
- I3G4250D_ID
 - I3g4250d_Infos, [132](#)
- I3g4250d_Infos, [132](#)
 - I3G4250D_I2C_ADD_H, [132](#)
 - I3G4250D_I2C_ADD_L, [132](#)
 - I3G4250D_ID, [132](#)
- I3G4250D_INT1_CFG
 - I3G4250D, [14](#)
- i3g4250d_int1_cfg_t, [149](#)
 - and_or, [149](#)
 - lir, [149](#)
 - xhie, [149](#)
 - xlir, [150](#)
 - yhie, [150](#)
 - ylie, [150](#)
 - zhie, [150](#)
 - zlie, [150](#)
- I3G4250D_INT1_DURATION
 - I3G4250D, [14](#)
- i3g4250d_int1_duration_t, [150](#)
 - d, [150](#)
 - wait, [150](#)
- I3G4250D_INT1_ON_TH_AND
 - I3G4250D, [16](#)
- I3G4250D_INT1_ON_TH_OR
 - I3G4250D, [16](#)
- i3g4250d_int1_route_t, [151](#)
 - i1_boot, [151](#)
 - i1_int1, [151](#)
- i3g4250d_int1_sel_t
 - I3G4250D, [19](#)
- I3G4250D_INT1_SRC
 - I3G4250D, [14](#)
- i3g4250d_int1_src_t, [151](#)
 - ia, [152](#)
 - not_used_01, [152](#)
 - xh, [152](#)
 - xl, [152](#)
 - yh, [152](#)
 - yl, [152](#)
 - zh, [152](#)
 - zl, [152](#)
- I3G4250D_INT1_TSH_XH
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_xh_t, [153](#)
 - not_used_01, [153](#)
 - thsx, [153](#)
- I3G4250D_INT1_TSH_XL
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_xl_t, [153](#)
 - thsx, [154](#)
- I3G4250D_INT1_TSH_YH
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_yh_t, [154](#)
 - not_used_01, [154](#)
 - thsy, [154](#)
- I3G4250D_INT1_TSH_YL
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_yl_t, [154](#)
 - thsy, [155](#)
- I3G4250D_INT1_TSH_ZH
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_zh_t, [155](#)
 - not_used_01, [155](#)
 - thsz, [155](#)
- I3G4250D_INT1_TSH_ZL
 - I3G4250D, [15](#)
- i3g4250d_int1_tsh_zl_t, [155](#)
 - thsz, [156](#)
- i3g4250d_int2_route_t, [156](#)
 - i2_drdy, [156](#)
 - i2_empty, [156](#)
 - i2_orun, [156](#)
 - i2_wtm, [156](#)
- I3G4250D_INT_LATCHED
 - I3G4250D, [19](#)
- i3g4250d_int_notification_get
 - I3G4250D, [46](#)
 - I3G4250D_interrupt_pins, [104](#)
- i3g4250d_int_notification_set
 - I3G4250D, [46](#)
 - I3G4250D_interrupt_pins, [104](#)
- i3g4250d_int_on_threshold_conf_get
 - I3G4250D, [48](#)
 - interrupt_on_threshold, [112](#)
- i3g4250d_int_on_threshold_conf_set
 - I3G4250D, [49](#)
 - interrupt_on_threshold, [113](#)
- i3g4250d_int_on_threshold_dur_get
 - I3G4250D, [49](#)
 - interrupt_on_threshold, [113](#)
- i3g4250d_int_on_threshold_dur_set
 - I3G4250D, [50](#)
 - interrupt_on_threshold, [114](#)
- i3g4250d_int_on_threshold_mode_get
 - I3G4250D, [51](#)
 - interrupt_on_threshold, [115](#)
- i3g4250d_int_on_threshold_mode_set
 - I3G4250D, [51](#)
 - interrupt_on_threshold, [115](#)

i3g4250d_int_on_threshold_src_get
 I3G4250D, 52
 interrupt_on_threshold, 116
 I3G4250D_INT_PULSED
 I3G4250D, 19
 i3g4250d_int_x_threshold_get
 I3G4250D, 53
 interrupt_on_threshold, 117
 i3g4250d_int_x_threshold_set
 I3G4250D, 53
 interrupt_on_threshold, 117
 i3g4250d_int_y_threshold_get
 I3G4250D, 54
 interrupt_on_threshold, 118
 i3g4250d_int_y_threshold_set
 I3G4250D, 55
 interrupt_on_threshold, 119
 i3g4250d_int_z_threshold_get
 I3G4250D, 55
 interrupt_on_threshold, 119
 i3g4250d_int_z_threshold_set
 I3G4250D, 56
 interrupt_on_threshold, 120
 I3G4250D_Interfaces_Functions, 71
 i3g4250d_read_reg, 71
 i3g4250d_write_reg, 72
 I3G4250D_interrupt_pins, 103
 i3g4250d_int_notification_get, 104
 i3g4250d_int_notification_set, 104
 i3g4250d_pin_int1_route_get, 105
 i3g4250d_pin_int1_route_set, 105
 i3g4250d_pin_int2_route_get, 106
 i3g4250d_pin_int2_route_set, 107
 i3g4250d_pin_mode_get, 107
 i3g4250d_pin_mode_set, 108
 i3g4250d_pin_polarity_get, 109
 i3g4250d_pin_polarity_set, 109
 i3g4250d_lir_t
 I3G4250D, 19
 i3g4250d_lp_bandwidth_get
 I3G4250D, 57
 I3G4250D_filters, 99
 i3g4250d_lp_bandwidth_set
 I3G4250D, 57
 I3G4250D_filters, 100
 I3G4250D_LPF1_HP_LPF2_ON_INT
 I3G4250D, 19
 I3G4250D_LPF1_HP_LPF2_ON_OUT
 I3G4250D, 19
 I3G4250D_LPF1_HP_ON_INT
 I3G4250D, 19
 I3G4250D_LPF1_HP_ON_OUT
 I3G4250D, 19
 I3G4250D_LPF1_LPF2_ON_INT
 I3G4250D, 19
 I3G4250D_LPF1_LPF2_ON_OUT
 I3G4250D, 19
 I3G4250D_ODR_100Hz
 I3G4250D, 17
 I3G4250D_ODR_200Hz
 I3G4250D, 17
 I3G4250D_ODR_400Hz
 I3G4250D, 17
 I3G4250D_ODR_800Hz
 I3G4250D, 17
 I3G4250D_ODR_OFF
 I3G4250D, 17
 I3G4250D_ODR_SLEEP
 I3G4250D, 17
 I3G4250D_ONLY_LPF1_ON_INT
 I3G4250D, 19
 I3G4250D_ONLY_LPF1_ON_OUT
 I3G4250D, 19
 I3G4250D_OPEN_DRAIN
 I3G4250D, 20
 i3g4250d_out_sel_t
 I3G4250D, 19
 I3G4250D_OUT_TEMP
 I3G4250D, 15
 I3G4250D_OUT_X_H
 I3G4250D, 15
 I3G4250D_OUT_X_L
 I3G4250D, 15
 I3G4250D_OUT_Y_H
 I3G4250D, 15
 I3G4250D_OUT_Y_L
 I3G4250D, 16
 I3G4250D_OUT_Z_H
 I3G4250D, 16
 I3G4250D_OUT_Z_L
 I3G4250D, 16
 i3g4250d_pin_int1_route_get
 I3G4250D, 58
 I3G4250D_interrupt_pins, 105
 i3g4250d_pin_int1_route_set
 I3G4250D, 59
 I3G4250D_interrupt_pins, 105
 i3g4250d_pin_int2_route_get
 I3G4250D, 59
 I3G4250D_interrupt_pins, 106
 i3g4250d_pin_int2_route_set
 I3G4250D, 60
 I3G4250D_interrupt_pins, 107
 i3g4250d_pin_mode_get
 I3G4250D, 61
 I3G4250D_interrupt_pins, 107
 i3g4250d_pin_mode_set
 I3G4250D, 61
 I3G4250D_interrupt_pins, 108
 i3g4250d_pin_polarity_get
 I3G4250D, 63
 I3G4250D_interrupt_pins, 109
 i3g4250d_pin_polarity_set
 I3G4250D, 64
 I3G4250D_interrupt_pins, 109
 i3g4250d_pp_od_t

- I3G4250D, [19](#)
- I3G4250D_PUSH_PULL
 - I3G4250D, [20](#)
- i3g4250d_read_reg
 - I3G4250D, [64](#)
 - I3G4250D_Interfaces_Functions, [71](#)
- I3G4250D_REFERENCE
 - I3G4250D, [16](#)
- i3g4250d_reference_t, [157](#)
 - ref, [157](#)
- i3g4250d_reg.c, [203](#)
- i3g4250d_reg.h, [165](#)
- i3g4250d_reg_t, [157](#)
 - bitwise, [159](#)
 - byte, [159](#)
 - ctrl_reg1, [159](#)
 - ctrl_reg2, [159](#)
 - ctrl_reg3, [159](#)
 - ctrl_reg4, [159](#)
 - ctrl_reg5, [159](#)
 - fifo_ctrl_reg, [159](#)
 - fifo_src_reg, [160](#)
 - int1_cfg, [160](#)
 - int1_duration, [160](#)
 - int1_src, [160](#)
 - int1_tsh_xh, [160](#)
 - int1_tsh_xl, [160](#)
 - int1_tsh_yh, [160](#)
 - int1_tsh_yl, [160](#)
 - int1_tsh_zh, [160](#)
 - int1_tsh_zl, [160](#)
 - reference, [161](#)
 - status_reg, [161](#)
- i3g4250d_self_test_get
 - I3G4250D, [65](#)
 - I3G4250D_common, [89](#)
- i3g4250d_self_test_set
 - I3G4250D, [66](#)
 - I3G4250D_common, [89](#)
- I3G4250D_Sensitivity, [73](#)
 - i3g4250d_from_fs245dps_to_mdps, [74](#)
 - i3g4250d_from_lsb_to_celsius, [74](#)
- I3G4250D_serial_interface, [101](#)
 - i3g4250d_spi_mode_get, [101](#)
 - i3g4250d_spi_mode_set, [102](#)
- i3g4250d_sim_t
 - I3G4250D, [20](#)
- I3G4250D_SPI_3_WIRE
 - I3G4250D, [20](#)
- I3G4250D_SPI_4_WIRE
 - I3G4250D, [20](#)
- i3g4250d_spi_mode_get
 - I3G4250D, [66](#)
 - I3G4250D_serial_interface, [101](#)
- i3g4250d_spi_mode_set
 - I3G4250D, [67](#)
 - I3G4250D_serial_interface, [102](#)
- i3g4250d_st_t
 - I3G4250D, [20](#)
- I3G4250D_STATUS_REG
 - I3G4250D, [16](#)
- i3g4250d_status_reg_get
 - I3G4250D, [68](#)
 - I3G4250D_data_generation, [79](#)
- i3g4250d_status_reg_t, [161](#)
 - _xor, [161](#)
 - xda, [161](#)
 - yda, [161](#)
 - yor, [161](#)
 - zda, [162](#)
 - zor, [162](#)
 - zyxda, [162](#)
 - zyxor, [162](#)
- i3g4250d_temperature_raw_get
 - I3G4250D, [68](#)
 - I3G4250D_Dataoutput, [82](#)
- I3G4250D_WHO_AM_I
 - I3G4250D, [16](#)
- i3g4250d_write_reg
 - I3G4250D, [69](#)
 - I3G4250D_Interfaces_Functions, [72](#)
- ia
 - i3g4250d_int1_src_t, [152](#)
- initialise_monitor_handles
 - syscalls.c, [221](#)
- INSTRUCTION_CACHE_ENABLE
 - stm32f4xx_hal_conf.h, [186](#)
- int1_cfg
 - i3g4250d_reg_t, [160](#)
- int1_duration
 - i3g4250d_reg_t, [160](#)
- int1_sel
 - i3g4250d_ctrl_reg5_t, [147](#)
- int1_src
 - i3g4250d_reg_t, [160](#)
- int1_tsh_xh
 - i3g4250d_reg_t, [160](#)
- int1_tsh_xl
 - i3g4250d_reg_t, [160](#)
- int1_tsh_yh
 - i3g4250d_reg_t, [160](#)
- int1_tsh_yl
 - i3g4250d_reg_t, [160](#)
- int1_tsh_zh
 - i3g4250d_reg_t, [160](#)
- int1_tsh_zl
 - i3g4250d_reg_t, [160](#)
- Interfaces_Functions, [131](#)
 - stmdev_mdelay_ptr, [131](#)
 - stmdev_read_ptr, [131](#)
 - stmdev_write_ptr, [131](#)
- interrupt_on_threshold, [111](#)
 - i3g4250d_int_on_threshold_conf_get, [112](#)
 - i3g4250d_int_on_threshold_conf_set, [113](#)
 - i3g4250d_int_on_threshold_dur_get, [113](#)
 - i3g4250d_int_on_threshold_dur_set, [114](#)

- i3g4250d_int_on_threshold_mode_get, [115](#)
- i3g4250d_int_on_threshold_mode_set, [115](#)
- i3g4250d_int_on_threshold_src_get, [116](#)
- i3g4250d_int_x_threshold_get, [117](#)
- i3g4250d_int_x_threshold_set, [117](#)
- i3g4250d_int_y_threshold_get, [118](#)
- i3g4250d_int_y_threshold_set, [119](#)
- i3g4250d_int_z_threshold_get, [119](#)
- i3g4250d_int_z_threshold_set, [120](#)
- lir
 - i3g4250d_int1_cfg_t, [149](#)
- LSE_STARTUP_TIMEOUT
 - stm32f4xx_hal_conf.h, [186](#)
- LSE_VALUE
 - stm32f4xx_hal_conf.h, [186](#)
- LSI_VALUE
 - stm32f4xx_hal_conf.h, [186](#)
- LSM9DS1_Register_Union, [132](#)
- MAC_ADDR0
 - stm32f4xx_hal_conf.h, [186](#)
- MAC_ADDR1
 - stm32f4xx_hal_conf.h, [187](#)
- MAC_ADDR2
 - stm32f4xx_hal_conf.h, [187](#)
- MAC_ADDR3
 - stm32f4xx_hal_conf.h, [187](#)
- MAC_ADDR4
 - stm32f4xx_hal_conf.h, [187](#)
- MAC_ADDR5
 - stm32f4xx_hal_conf.h, [187](#)
- main
 - main.c, [208](#)
- main.c, [207](#)
 - BOOT_TIME, [208](#)
 - hspi1, [211](#)
 - huart2, [211](#)
 - main, [208](#)
 - SystemClock_Config, [210](#)
- main.h, [179](#)
 - Error_Handler, [180](#)
 - GYRO_CS_OUT_GPIO_Port, [180](#)
 - GYRO_CS_OUT_Pin, [180](#)
- mdelay
 - stmdev_ctx_t, [162](#)
- MemManage_Handler
 - stm32f4xx_it.c, [215](#)
 - stm32f4xx_it.h, [201](#)
- MEMS_SHARED_TYPES
 - sensors common types, [130](#)
- MEMS_UCF_SHARED_TYPES
 - sensors common types, [130](#)
- NMI_Handler
 - stm32f4xx_it.c, [215](#)
 - stm32f4xx_it.h, [201](#)
- not_used_01
 - i3g4250d_ctrl_reg2_t, [143](#)
- i3g4250d_ctrl_reg4_t, [146](#)
- i3g4250d_ctrl_reg5_t, [147](#)
- i3g4250d_int1_src_t, [152](#)
- i3g4250d_int1_tsh_xh_t, [153](#)
- i3g4250d_int1_tsh_yh_t, [154](#)
- i3g4250d_int1_tsh_zh_t, [155](#)
- not_used_02
 - i3g4250d_ctrl_reg4_t, [146](#)
- out_sel
 - i3g4250d_ctrl_reg5_t, [147](#)
- ovrn
 - i3g4250d_fifo_src_reg_t, [148](#)
- pd
 - i3g4250d_ctrl_reg1_t, [143](#)
- PendSV_Handler
 - stm32f4xx_it.c, [215](#)
 - stm32f4xx_it.h, [201](#)
- PHY_AUTONEGO_COMPLETE
 - stm32f4xx_hal_conf.h, [187](#)
- PHY_AUTONEGOTIATION
 - stm32f4xx_hal_conf.h, [187](#)
- PHY_BCR
 - stm32f4xx_hal_conf.h, [187](#)
- PHY_BSR
 - stm32f4xx_hal_conf.h, [187](#)
- PHY_CONFIG_DELAY
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_DUPLEX_STATUS
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_FULLDUPLEX_100M
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_FULLDUPLEX_10M
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_HALFDUPLEX_100M
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_HALFDUPLEX_10M
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_ISOLATE
 - stm32f4xx_hal_conf.h, [188](#)
- PHY_JABBER_DETECTION
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_LINKED_STATUS
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_LOOPBACK
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_POWERDOWN
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_READ_TO
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_RESET
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_RESET_DELAY
 - stm32f4xx_hal_conf.h, [189](#)
- PHY_RESTART_AUTONEGOTIATION
 - stm32f4xx_hal_conf.h, [190](#)
- PHY_SPEED_STATUS
 - stm32f4xx_hal_conf.h, [190](#)

- PHY_SR
 - stm32f4xx_hal_conf.h, 190
- PHY_WRITE_TO
 - stm32f4xx_hal_conf.h, 190
- pp_od
 - i3g4250d_ctrl_reg3_t, 145
- PREFETCH_ENABLE
 - stm32f4xx_hal_conf.h, 190
- PROPERTY_DISABLE
 - sensors common types, 130
- PROPERTY_ENABLE
 - sensors common types, 130
- read_reg
 - stmdev_ctx_t, 163
- ref
 - i3g4250d_reference_t, 157
- reference
 - i3g4250d_reg_t, 161
- sensors common types, 129
 - MEMS_SHARED_TYPES, 130
 - MEMS_UCF_SHARED_TYPES, 130
 - PROPERTY_DISABLE, 130
 - PROPERTY_ENABLE, 130
- sim
 - i3g4250d_ctrl_reg4_t, 146
- st
 - i3g4250d_ctrl_reg4_t, 146
- status_reg
 - i3g4250d_reg_t, 161
- stm32f4xx_hal_conf.h, 181
 - assert_param, 183
 - DATA_CACHE_ENABLE, 183
 - DP83848_PHY_ADDRESS, 184
 - ETH_RX_BUF_SIZE, 184
 - ETH_RXBUFNB, 184
 - ETH_TX_BUF_SIZE, 184
 - ETH_TXBUFNB, 184
 - EXTERNAL_CLOCK_VALUE, 184
 - HAL_CORTEX_MODULE_ENABLED, 184
 - HAL_DMA_MODULE_ENABLED, 184
 - HAL_EXTI_MODULE_ENABLED, 184
 - HAL_FLASH_MODULE_ENABLED, 185
 - HAL_GPIO_MODULE_ENABLED, 185
 - HAL_MODULE_ENABLED, 185
 - HAL_PWR_MODULE_ENABLED, 185
 - HAL_RCC_MODULE_ENABLED, 185
 - HAL_SPI_MODULE_ENABLED, 185
 - HAL_UART_MODULE_ENABLED, 185
 - HSE_STARTUP_TIMEOUT, 185
 - HSE_VALUE, 185
 - HSI_VALUE, 186
 - INSTRUCTION_CACHE_ENABLE, 186
 - LSE_STARTUP_TIMEOUT, 186
 - LSE_VALUE, 186
 - LSI_VALUE, 186
 - MAC_ADDR0, 186
 - MAC_ADDR1, 187
 - MAC_ADDR2, 187
 - MAC_ADDR3, 187
 - MAC_ADDR4, 187
 - MAC_ADDR5, 187
 - PHY_AUTONEGO_COMPLETE, 187
 - PHY_AUTONEGOTIATION, 187
 - PHY_BCR, 187
 - PHY_BSR, 187
 - PHY_CONFIG_DELAY, 188
 - PHY_DUPLEX_STATUS, 188
 - PHY_FULLDUPLEX_100M, 188
 - PHY_FULLDUPLEX_10M, 188
 - PHY_HALFDUPLEX_100M, 188
 - PHY_HALFDUPLEX_10M, 188
 - PHY_ISOLATE, 188
 - PHY_JABBER_DETECTION, 189
 - PHY_LINKED_STATUS, 189
 - PHY_LOOPBACK, 189
 - PHY_POWERDOWN, 189
 - PHY_READ_TO, 189
 - PHY_RESET, 189
 - PHY_RESET_DELAY, 189
 - PHY_RESTART_AUTONEGOTIATION, 190
 - PHY_SPEED_STATUS, 190
 - PHY_SR, 190
 - PHY_WRITE_TO, 190
 - PREFETCH_ENABLE, 190
 - TICK_INT_PRIORITY, 190
 - USE_HAL_ADC_REGISTER_CALLBACKS, 190
 - USE_HAL_CAN_REGISTER_CALLBACKS, 190
 - USE_HAL_CEC_REGISTER_CALLBACKS, 191
 - USE_HAL_CRYPT_REGISTER_CALLBACKS, 191
 - USE_HAL_DAC_REGISTER_CALLBACKS, 191
 - USE_HAL_DCMI_REGISTER_CALLBACKS, 191
 - USE_HAL_DFSDM_REGISTER_CALLBACKS, 191
 - USE_HAL_DMA2D_REGISTER_CALLBACKS, 191
 - USE_HAL_DSI_REGISTER_CALLBACKS, 191
 - USE_HAL_ETH_REGISTER_CALLBACKS, 191
 - USE_HAL_FMPI2C_REGISTER_CALLBACKS, 191
 - USE_HAL_FMPMBUS_REGISTER_CALLBACKS, 191
 - USE_HAL_HASH_REGISTER_CALLBACKS, 192
 - USE_HAL_HCD_REGISTER_CALLBACKS, 192
 - USE_HAL_I2C_REGISTER_CALLBACKS, 192
 - USE_HAL_I2S_REGISTER_CALLBACKS, 192
 - USE_HAL_IRDA_REGISTER_CALLBACKS, 192
 - USE_HAL_LPTIM_REGISTER_CALLBACKS, 192
 - USE_HAL_LTDC_REGISTER_CALLBACKS, 192
 - USE_HAL_MMC_REGISTER_CALLBACKS, 192
 - USE_HAL_NAND_REGISTER_CALLBACKS, 192
 - USE_HAL_NOR_REGISTER_CALLBACKS, 192
 - USE_HAL_PCCARD_REGISTER_CALLBACKS, 193
 - USE_HAL_PCD_REGISTER_CALLBACKS, 193
 - USE_HAL_QSPI_REGISTER_CALLBACKS, 193

- USE_HAL_RNG_REGISTER_CALLBACKS, 193
- USE_HAL_RTC_REGISTER_CALLBACKS, 193
- USE_HAL_SAI_REGISTER_CALLBACKS, 193
- USE_HAL_SD_REGISTER_CALLBACKS, 193
- USE_HAL_SDRAM_REGISTER_CALLBACKS, 193
- USE_HAL_SMARTCARD_REGISTER_CALLBACKS, 193
- USE_HAL_SMBUS_REGISTER_CALLBACKS, 193
- USE_HAL_SPDIFRX_REGISTER_CALLBACKS, 194
- USE_HAL_SPI_REGISTER_CALLBACKS, 194
- USE_HAL_SRAM_REGISTER_CALLBACKS, 194
- USE_HAL_TIM_REGISTER_CALLBACKS, 194
- USE_HAL_UART_REGISTER_CALLBACKS, 194
- USE_HAL_USART_REGISTER_CALLBACKS, 194
- USE_HAL_WWDG_REGISTER_CALLBACKS, 194
- USE_RTOS, 194
- USE_SPI_CRC, 194
- VDD_VALUE, 194
- stm32f4xx_hal_msp.c, 211
 - HAL_MspInit, 212
 - HAL_SPI_MspDeInit, 212
 - HAL_SPI_MspInit, 213
 - HAL_UART_MspDeInit, 213
 - HAL_UART_MspInit, 213
- stm32f4xx_it.c, 214
 - BusFault_Handler, 215
 - DebugMon_Handler, 215
 - HardFault_Handler, 215
 - MemManage_Handler, 215
 - NMI_Handler, 215
 - PendSV_Handler, 215
 - SVC_Handler, 216
 - SysTick_Handler, 216
 - UsageFault_Handler, 216
- stm32f4xx_it.h, 200
 - BusFault_Handler, 201
 - DebugMon_Handler, 201
 - HardFault_Handler, 201
 - MemManage_Handler, 201
 - NMI_Handler, 201
 - PendSV_Handler, 201
 - SVC_Handler, 202
 - SysTick_Handler, 202
 - UsageFault_Handler, 202
- Stm32f4xx_system, 134
- STM32F4xx_System_Private_Defines, 135
- STM32F4xx_System_Private_FunctionPrototypes, 137
- STM32F4xx_System_Private_Functions, 137
 - SystemCoreClockUpdate, 138
 - SystemInit, 139
- STM32F4xx_System_Private_Includes, 134
 - HSE_VALUE, 135
 - HSI_VALUE, 135
- STM32F4xx_System_Private_Macros, 136
- STM32F4xx_System_Private_TypesDefinitions, 135
- STM32F4xx_System_Private_Variables, 136
 - AHBPrescTable, 136
 - APBPrescTable, 136
 - SystemCoreClock, 136
- stmdev_ctx_t, 162
 - handle, 162
 - mdelay, 162
 - read_reg, 163
 - write_reg, 163
- stmdev_mdelay_ptr
 - Interfaces_Functions, 131
- stmdev_read_ptr
 - Interfaces_Functions, 131
- stmdev_write_ptr
 - Interfaces_Functions, 131
- SVC_Handler
 - stm32f4xx_it.c, 216
 - stm32f4xx_it.h, 202
- syscalls.c, 216
 - __attribute__, 218
 - __io_getchar, 218
 - __io_putchar, 218
 - _close, 218
 - _execve, 218
 - _exit, 219
 - _fork, 219
 - _fstat, 219
 - _getpid, 219
 - _isatty, 219
 - _kill, 219
 - _link, 220
 - _lseek, 220
 - _open, 220
 - _stat, 220
 - _times, 220
 - _unlink, 220
 - _wait, 221
 - environ, 221
 - initialise_monitor_handles, 221
- systemem.c, 221
 - _sbrk, 222
- system_stm32f4xx.c, 223
- SystemClock_Config
 - main.c, 210
- SystemCoreClock
 - STM32F4xx_System_Private_Variables, 136
- SystemCoreClockUpdate
 - STM32F4xx_System_Private_Functions, 138
- SystemInit
 - STM32F4xx_System_Private_Functions, 139
- SysTick_Handler
 - stm32f4xx_it.c, 216
 - stm32f4xx_it.h, 202
- thsx
 - i3g4250d_int1_tsh_xh_t, 153
 - i3g4250d_int1_tsh_xl_t, 154

- thsy
 - i3g4250d_int1_tsh_yh_t, [154](#)
 - i3g4250d_int1_tsh_yl_t, [155](#)
- thsz
 - i3g4250d_int1_tsh_zh_t, [155](#)
 - i3g4250d_int1_tsh_zl_t, [156](#)
- TICK_INT_PRIORITY
 - stm32f4xx_hal_conf.h, [190](#)
- ucf_line_t, [163](#)
 - address, [163](#)
 - data, [163](#)
- UsageFault_Handler
 - stm32f4xx_it.c, [216](#)
 - stm32f4xx_it.h, [202](#)
- USE_HAL_ADC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [190](#)
- USE_HAL_CAN_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [190](#)
- USE_HAL_CEC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_Cryp_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_DAC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_DCMI_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_DFSDM_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_DMA2D_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_DSI_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_ETH_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_FMPI2C_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_FMPSPMBUS_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [191](#)
- USE_HAL_HASH_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_HCD_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_I2C_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_I2S_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_IRDA_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_LPTIM_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_LTDC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_MMC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_NAND_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_NOR_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [192](#)
- USE_HAL_PCCARD_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_PCD_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_QSPI_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_RNG_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_RTC_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SAI_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SD_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SDRAM_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SMARTCARD_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SMBUS_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [193](#)
- USE_HAL_SPDIFRX_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_SPI_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_SRAM_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_TIM_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_UART_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_USART_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_HAL_WWDG_REGISTER_CALLBACKS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_RTOS
 - stm32f4xx_hal_conf.h, [194](#)
- USE_SPI_CRC
 - stm32f4xx_hal_conf.h, [194](#)
- VDD_VALUE
 - stm32f4xx_hal_conf.h, [194](#)
- wait
 - i3g4250d_int1_duration_t, [150](#)
- write_reg
 - stmdev_ctx_t, [163](#)
- wtm
 - i3g4250d_fifo_ctrl_reg_t, [148](#)
 - i3g4250d_fifo_src_reg_t, [149](#)
- xda
 - i3g4250d_status_reg_t, [161](#)
- xh
 - i3g4250d_int1_src_t, [152](#)
- xhie
 - i3g4250d_int1_cfg_t, [149](#)
- xl
 - i3g4250d_int1_src_t, [152](#)
- xlie

i3g4250d_int1_cfg_t, [150](#)

yda
i3g4250d_status_reg_t, [161](#)

yh
i3g4250d_int1_src_t, [152](#)

yhie
i3g4250d_int1_cfg_t, [150](#)

yl
i3g4250d_int1_src_t, [152](#)

ylie
i3g4250d_int1_cfg_t, [150](#)

yor
i3g4250d_status_reg_t, [161](#)

zda
i3g4250d_status_reg_t, [162](#)

zh
i3g4250d_int1_src_t, [152](#)

zhie
i3g4250d_int1_cfg_t, [150](#)

zl
i3g4250d_int1_src_t, [152](#)

zlie
i3g4250d_int1_cfg_t, [150](#)

zor
i3g4250d_status_reg_t, [162](#)

zyxda
i3g4250d_status_reg_t, [162](#)

zyxor
i3g4250d_status_reg_t, [162](#)