

# eMD ICM426xx Driver API

## 2.0.1

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
4.1	Icm426xx driver high level functions . . . . .	7
4.1.1	Detailed Description . . . . .	10
4.1.2	Macro Definition Documentation . . . . .	10
4.1.2.1	ICM426XX_FIFO_MIRRORING_SIZE . . . . .	10
4.1.2.2	INV_ICM426XX_LIGHTWEIGHT_DRIVER . . . . .	10
4.1.3	Enumeration Type Documentation . . . . .	10
4.1.3.1	INV_ICM426XX_FIFO_CONFIG_t . . . . .	10
4.1.3.2	inv_icm426xx_sensor . . . . .	11
4.1.4	Function Documentation . . . . .	11
4.1.4.1	inv_icm426xx_configure_fifo(struct inv_icm426xx *s, INV_ICM426XX_FIFO_CONFIG_t fifo_config) . . . . .	11
4.1.4.2	inv_icm426xx_configure_fifo_wm(struct inv_icm426xx *s, uint16_t wm) . . . . .	11
4.1.4.3	inv_icm426xx_configure_timestamp_resolution(struct inv_icm426xx *s, ICM426XX_TMST_CONFIG_RESOL_t resol) . . . . .	12
4.1.4.4	inv_icm426xx_convert_odr_bitfield_to_us(uint32_t odr_bitfield) . . . . .	12
4.1.4.5	inv_icm426xx_device_reset(struct inv_icm426xx *s) . . . . .	12

4.1.4.6	<code>inv_icm426xx_disable_accel(struct inv_icm426xx *s)</code>	12
4.1.4.7	<code>inv_icm426xx_disable_fsync(struct inv_icm426xx *s)</code>	13
4.1.4.8	<code>inv_icm426xx_disable_gyro(struct inv_icm426xx *s)</code>	13
4.1.4.9	<code>inv_icm426xx_disable_high_resolution_fifo(struct inv_icm426xx *s)</code>	13
4.1.4.10	<code>inv_icm426xx_disable_timestamp_to_register(struct inv_icm426xx *s)</code>	13
4.1.4.11	<code>inv_icm426xx_enable_accel_low_noise_mode(struct inv_icm426xx *s)</code>	14
4.1.4.12	<code>inv_icm426xx_enable_accel_low_power_mode(struct inv_icm426xx *s)</code>	14
4.1.4.13	<code>inv_icm426xx_enable_clkin_rtc(struct inv_icm426xx *s, uint8_t enable)</code>	14
4.1.4.14	<code>inv_icm426xx_enable_fsync(struct inv_icm426xx *s)</code>	15
4.1.4.15	<code>inv_icm426xx_enable_gyro_low_noise_mode(struct inv_icm426xx *s)</code>	15
4.1.4.16	<code>inv_icm426xx_enable_high_resolution_fifo(struct inv_icm426xx *s)</code>	15
4.1.4.17	<code>inv_icm426xx_enable_timestamp_to_register(struct inv_icm426xx *s)</code>	15
4.1.4.18	<code>inv_icm426xx_force_clock_source(struct inv_icm426xx *s, ICM426XX_INTF_↵ CONFIG1_ACCEL_LP_CLK_t clk_src)</code>	15
4.1.4.19	<code>inv_icm426xx_get_accel_fsr(struct inv_icm426xx *s, ICM426XX_ACCEL_CO↵ NFIG0_FS_SEL_t *accel_fsr_g)</code>	16
4.1.4.20	<code>inv_icm426xx_get_clkin_rtc_status(struct inv_icm426xx *s)</code>	16
4.1.4.21	<code>inv_icm426xx_get_config_ibi(struct inv_icm426xx *s, inv_icm426xx_interrupt_↵ parameter_t *interrupt_to_configure)</code>	16
4.1.4.22	<code>inv_icm426xx_get_config_int1(struct inv_icm426xx *s, inv_icm426xx_interrupt_↵ _parameter_t *interrupt_to_configure)</code>	17
4.1.4.23	<code>inv_icm426xx_get_config_int2(struct inv_icm426xx *s, inv_icm426xx_interrupt_↵ _parameter_t *interrupt_to_configure)</code>	17
4.1.4.24	<code>inv_icm426xx_get_current_timestamp(struct inv_icm426xx *s, uint32_t *icm_time)</code>	17
4.1.4.25	<code>inv_icm426xx_get_data_from_fifo(struct inv_icm426xx *s)</code>	18
4.1.4.26	<code>inv_icm426xx_get_data_from_registers(struct inv_icm426xx *s)</code>	18
4.1.4.27	<code>inv_icm426xx_get_fifo_timestamp_resolution_us_q24(struct inv_icm426xx *s)</code>	18
4.1.4.28	<code>inv_icm426xx_get_gyro_fsr(struct inv_icm426xx *s, ICM426XX_GYRO_CONF↵ IG0_FS_SEL_t *gyro_fsr_dps)</code>	18
4.1.4.29	<code>inv_icm426xx_get_reg_timestamp_resolution_us_q24(struct inv_icm426xx *s)</code>	19
4.1.4.30	<code>inv_icm426xx_get_version(void)</code>	19
4.1.4.31	<code>inv_icm426xx_get_who_am_i(struct inv_icm426xx *s, uint8_t *who_am_i)</code>	19

4.1.4.32	<code>inv_icm426xx_init(struct inv_icm426xx *s, struct inv_icm426xx_serif *serif, void(*sensor_event_cb)(inv_icm426xx_sensor_event_t *event))</code> . . . . .	19
4.1.4.33	<code>inv_icm426xx_reset_fifo(struct inv_icm426xx *s)</code> . . . . .	20
4.1.4.34	<code>inv_icm426xx_set_accel_frequency(struct inv_icm426xx *s, const ICM426XX_ACCEL_CONFIG0_ODR_t frequency)</code> . . . . .	20
4.1.4.35	<code>inv_icm426xx_set_accel_fsr(struct inv_icm426xx *s, ICM426XX_ACCEL_CONFIG0_FS_SEL_t accel_fsr_g)</code> . . . . .	21
4.1.4.36	<code>inv_icm426xx_set_accel_ln_bw(struct inv_icm426xx *s, ICM426XX_GYRO_CONFIG0_ACCEL_CONFIG0_ACCEL_FILT_BW_t acc_bw)</code> . . . . .	21
4.1.4.37	<code>inv_icm426xx_set_accel_lp_avg(struct inv_icm426xx *s, ICM426XX_GYRO_CONFIG0_ACCEL_CONFIG0_ACCEL_FILT_AVG_t acc_avg)</code> . . . . .	22
4.1.4.38	<code>inv_icm426xx_set_config_ibi(struct inv_icm426xx *s, inv_icm426xx_interrupt_parameter_t *interrupt_to_configure)</code> . . . . .	22
4.1.4.39	<code>inv_icm426xx_set_config_int1(struct inv_icm426xx *s, inv_icm426xx_interrupt_parameter_t *interrupt_to_configure)</code> . . . . .	22
4.1.4.40	<code>inv_icm426xx_set_config_int2(struct inv_icm426xx *s, inv_icm426xx_interrupt_parameter_t *interrupt_to_configure)</code> . . . . .	23
4.1.4.41	<code>inv_icm426xx_set_gyro_frequency(struct inv_icm426xx *s, const ICM426XX_GYRO_CONFIG0_ODR_t frequency)</code> . . . . .	23
4.1.4.42	<code>inv_icm426xx_set_gyro_fsr(struct inv_icm426xx *s, ICM426XX_GYRO_CONFIG0_FS_SEL_t gyro_fsr_dps)</code> . . . . .	23
4.1.4.43	<code>inv_icm426xx_set_gyro_ln_bw(struct inv_icm426xx *s, ICM426XX_GYRO_CONFIG0_GYRO_FILT_BW_t gyr_bw)</code> . . . . .	24
4.1.4.44	<code>inv_icm426xx_set_reg_bank(struct inv_icm426xx *s, uint8_t bank)</code> . . . . .	24
4.2	Icm426xx driver high level functions related to APEX . . . . .	25
4.2.1	Detailed Description . . . . .	26
4.2.2	Function Documentation . . . . .	26
4.2.2.1	<code>inv_icm426xx_configure_apex_parameters(struct inv_icm426xx *s, const inv_icm426xx_apex_parameters_t *apex_inputs)</code> . . . . .	26
4.2.2.2	<code>inv_icm426xx_configure_smd_wom(struct inv_icm426xx *s, const uint8_t x_th, const uint8_t y_th, const uint8_t z_th, ICM426XX_SMD_CONFIG_WOM_INT_MODE_t wom_int, ICM426XX_SMD_CONFIG_WOM_MODE_t wom_mode)</code> . . . . .	27
4.2.2.3	<code>inv_icm426xx_configure_tap_parameters(struct inv_icm426xx *s, const inv_icm426xx_tap_parameters_t *tap_inputs)</code> . . . . .	27
4.2.2.4	<code>inv_icm426xx_disable_apex_pedometer(struct inv_icm426xx *s)</code> . . . . .	28
4.2.2.5	<code>inv_icm426xx_disable_apex_r2w(struct inv_icm426xx *s)</code> . . . . .	28
4.2.2.6	<code>inv_icm426xx_disable_apex_tilt(struct inv_icm426xx *s)</code> . . . . .	28

4.2.2.7	inv_icm426xx_disable_smd(struct inv_icm426xx *s)	28
4.2.2.8	inv_icm426xx_disable_tap(struct inv_icm426xx *s)	29
4.2.2.9	inv_icm426xx_disable_wom(struct inv_icm426xx *s)	29
4.2.2.10	inv_icm426xx_enable_apex_pedometer(struct inv_icm426xx *s)	29
4.2.2.11	inv_icm426xx_enable_apex_r2w(struct inv_icm426xx *s)	29
4.2.2.12	inv_icm426xx_enable_apex_tilt(struct inv_icm426xx *s)	30
4.2.2.13	inv_icm426xx_enable_smd(struct inv_icm426xx *s)	30
4.2.2.14	inv_icm426xx_enable_tap(struct inv_icm426xx *s)	30
4.2.2.15	inv_icm426xx_enable_wom(struct inv_icm426xx *s)	30
4.2.2.16	inv_icm426xx_get_apex_data_activity(struct inv_icm426xx *s, inv_icm426xx_↵ apex_step_activity_t *apex_activity)	30
4.2.2.17	inv_icm426xx_get_apex_parameters(struct inv_icm426xx *s, inv_icm426xx_↵ apex_parameters_t *apex_params)	31
4.2.2.18	inv_icm426xx_get_tap_data(struct inv_icm426xx *s, inv_icm426xx_tap_data_↵ t *tap_data)	31
4.2.2.19	inv_icm426xx_get_tap_parameters(struct inv_icm426xx *s, inv_icm426xx_tap_↵ _parameters_t *tap_params)	31
4.2.2.20	inv_icm426xx_init_apex_parameters_struct(struct inv_icm426xx *s, inv_↵ icm426xx_apex_parameters_t *apex_inputs)	32
4.2.2.21	inv_icm426xx_init_tap_parameters_struct(struct inv_icm426xx *s, inv_↵ icm426xx_tap_parameters_t *tap_inputs)	32
4.2.2.22	inv_icm426xx_set_apex_frequency(struct inv_icm426xx *s, const ICM426XX_↵ APEX_CONFIG0_DMP_ODR_t frequency)	32
4.2.2.23	inv_icm426xx_start_dmp(struct inv_icm426xx *s)	33
4.3	Icm426xx driver extern functions	34
4.3.1	Detailed Description	34
4.3.2	Function Documentation	34
4.3.2.1	inv_icm426xx_get_time_us(void)	34
4.3.2.2	inv_icm426xx_sleep_us(uint32_t us)	34
4.4	Icm426xx selftest	35
4.4.1	Detailed Description	35
4.4.2	Function Documentation	35
4.4.2.1	inv_icm426xx_get_st_bias(struct inv_icm426xx *s, int st_bias[6])	35
4.4.2.2	inv_icm426xx_run_selftest(struct inv_icm426xx *s, int *result)	35
4.4.2.3	inv_icm426xx_set_st_bias(struct inv_icm426xx *s, const int st_bias[6])	36
4.5	Icm426xx driver transport	37
4.5.1	Detailed Description	37
4.5.2	Function Documentation	37
4.5.2.1	inv_icm426xx_init_transport(struct inv_icm426xx *s)	37
4.5.2.2	inv_icm426xx_read_reg(struct inv_icm426xx *s, uint8_t reg, uint32_t len, uint8_↵ _t *buf)	37
4.5.2.3	inv_icm426xx_write_reg(struct inv_icm426xx *s, uint8_t reg, uint32_t len, const uint8_t *buf)	38

<b>5</b>	<b>Class Documentation</b>	<b>39</b>
5.1	<a href="#">fifo_header_t Union Reference</a>	39
5.1.1	Detailed Description	39
5.2	<a href="#">inv_icm426xx Struct Reference</a>	39
5.2.1	Detailed Description	41
5.2.2	Member Data Documentation	41
5.2.2.1	<a href="#">fifo_is_used</a>	41
5.2.2.2	<a href="#">sensor_event_cb</a>	41
5.3	<a href="#">inv_icm426xx_apex_parameters Struct Reference</a>	41
5.3.1	Detailed Description	42
5.3.2	Member Data Documentation	42
5.3.2.1	<a href="#">power_save</a>	42
5.3.2.2	<a href="#">r2w_gest_delay</a>	42
5.3.2.3	<a href="#">r2w_mounting_matrix</a>	43
5.3.2.4	<a href="#">r2w_sleep_time_out</a>	43
5.3.2.5	<a href="#">sensitivity_mode</a>	43
5.4	<a href="#">inv_icm426xx_apex_step_activity Struct Reference</a>	43
5.4.1	Detailed Description	43
5.4.2	Member Data Documentation	43
5.4.2.1	<a href="#">step_cadence</a>	43
5.5	<a href="#">inv_icm426xx_interrupt_parameter_t Struct Reference</a>	44
5.5.1	Detailed Description	44
5.6	<a href="#">inv_icm426xx_sensor_event_t Struct Reference</a>	44
5.6.1	Detailed Description	44
5.7	<a href="#">inv_icm426xx_serif Struct Reference</a>	44
5.7.1	Detailed Description	44
5.8	<a href="#">inv_icm426xx_tap_data Struct Reference</a>	45
5.8.1	Detailed Description	45
5.9	<a href="#">inv_icm426xx_tap_parameters_t Struct Reference</a>	45
5.9.1	Detailed Description	46
5.10	<a href="#">inv_icm426xx_transport Struct Reference</a>	46
5.10.1	Detailed Description	46
5.10.2	Member Data Documentation	47
5.10.2.1	<a href="#">register_cache</a>	47
5.11	<a href="#">recover_regs Struct Reference</a>	47
5.11.1	Detailed Description	47
5.12	<a href="#">inv_icm426xx_transport::register_cache Struct Reference</a>	47
5.12.1	Detailed Description	47

<b>6</b>	<b>File Documentation</b>	<b>49</b>
6.1	lcm426xxDefs.h File Reference	49
6.1.1	Detailed Description	50
6.1.2	Enumeration Type Documentation	50
6.1.2.1	ICM426XX_ACCEL_CONFIG0_FS_SEL_t	50
6.1.2.2	ICM426XX_ACCEL_CONFIG0_ODR_t	50
6.1.2.3	ICM426XX_APEX_CONFIG0_DMP_ODR_t	51
6.1.2.4	ICM426XX_GYRO_CONFIG0_FS_SEL_t	51
6.1.2.5	ICM426XX_GYRO_CONFIG0_ODR_t	51
6.1.2.6	ICM426XX_OIS1_CONFIG1_DEC_t	52
6.1.2.7	ICM426XX_OIS1_CONFIG2_ACCEL_FS_SEL_t	52
6.1.2.8	ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_t	52
6.1.2.9	ICM426XX_OIS2_CONFIG1_DEC_t	52
6.1.2.10	ICM426XX_OIS2_CONFIG2_ACCEL_FS_SEL_t	53
6.1.2.11	ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_t	53
6.2	lcm426xxDriver_HL.h File Reference	53
6.2.1	Detailed Description	56
6.3	lcm426xxDriver_HL_apex.h File Reference	57
6.3.1	Detailed Description	59
6.4	lcm426xxExtFunc.h File Reference	59
6.4.1	Detailed Description	59
6.5	lcm426xxSelfTest.h File Reference	60
6.5.1	Detailed Description	60
6.6	lcm426xxTransport.h File Reference	60
6.6.1	Detailed Description	61
	<b>Index</b>	<b>63</b>



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

lcm426xx driver high level functions . . . . .	7
lcm426xx driver high level functions related to APEX . . . . .	25
lcm426xx driver extern functions . . . . .	34
lcm426xx selftest . . . . .	35
lcm426xx driver transport . . . . .	37



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">fifo_header_t</a>	Describe the content of the FIFO header . . . . .	39
<a href="#">inv_icm426xx</a>	Icm426xx driver states definition . . . . .	39
<a href="#">inv_icm426xx_apex_parameters</a>	Icm426xx APEX inputs parameters definition . . . . .	41
<a href="#">inv_icm426xx_apex_step_activity</a>	APEX pedometer outputs . . . . .	43
<a href="#">inv_icm426xx_interrupt_parameter_t</a>	Icm426xx set of interrupt enable flag . . . . .	44
<a href="#">inv_icm426xx_sensor_event_t</a>	Sensor event structure definition . . . . .	44
<a href="#">inv_icm426xx_serif</a>	Basesensor serial interface . . . . .	44
<a href="#">inv_icm426xx_tap_data</a>	TAP outputs . . . . .	45
<a href="#">inv_icm426xx_tap_parameters_t</a>	Icm426xx TAP inputs parameters definition . . . . .	45
<a href="#">inv_icm426xx_transport</a>	Transport interface . . . . .	46
<a href="#">recover_regs</a>	Contains the current register values . . . . .	47
<a href="#">inv_icm426xx_transport::register_cache</a>	Contains mirrored values of some IP registers . . . . .	47



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">lcm426xxDefs.h</a>	
File exposing the device register map . . . . .	49
<a href="#">lcm426xxDriver_HL.h</a>	
High-level function to setup an lcm426xx device . . . . .	53
<a href="#">lcm426xxDriver_HL_apex.h</a>	
High-level function to setup an lcm426xx device . . . . .	57
<a href="#">lcm426xxExtFunc.h</a>	
Extern functions for lcm426xx devices . . . . .	59
<a href="#">lcm426xxSelfTest.h</a>	
Low-level function to run selftest on a lcm426xx device . . . . .	60
<a href="#">lcm426xxTransport.h</a>	
Low-level lcm426xx register access . . . . .	60



# Chapter 4

## Module Documentation

### 4.1 lcm426xx driver high level functions

High-level function to setup an lcm426xx device.

#### Files

- file [lcm426xxDriver\\_HL.h](#)  
*High-level function to setup an lcm426xx device.*

#### Classes

- struct [inv\\_icm426xx\\_sensor\\_event\\_t](#)  
*Sensor event structure definition.*
- struct [inv\\_icm426xx](#)  
*lcm426xx driver states definition.*
- struct [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#)  
*lcm426xx set of interrupt enable flag.*

#### Macros

- #define [INV\\_ICM426XX\\_LIGHTWEIGHT\\_DRIVER](#) 0  
*Lighthen driver logic by stripping out procedures on transitions.*
- #define [PLL\\_SCALE\\_FACTOR\\_Q24](#) (1UL<<24)  
*Scale factor and max ODR Dependant of chip.*
- #define [ACCEL\\_CONFIG0\\_FS\\_SEL\\_MAX](#) ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_16g  
*Max FSR values for accel and gyro Dependant of chip.*
- #define [RTC\\_SUPPORTED](#) 0  
*RTC Support flag Define whether the RTC mode is supported Dependant of chip.*
- #define [ICM426XX\\_FIFO\\_MIRRORING\\_SIZE](#) 16 \* 129  
*lcm426xx maximum buffer size mirrored from FIFO at polling time.*
- #define [ICM426XX\\_DEFAULT\\_WOM\\_THS\\_MG](#) 52>>2 /\* = 52mg/4 \*/  
*Default value for the WOM threshold Resolution of the threshold is ~ = 4mg.*
- #define [ICM426XX\\_ACC\\_STARTUP\\_TIME\\_US](#) 20000U  
*lcm426xx Accelerometer start-up time before having correct data.*
- #define [ICM426XX\\_GYR\\_STARTUP\\_TIME\\_US](#) 60000U  
*lcm426xx Gyroscope start-up time before having correct data.*

## Enumerations

## Functions

- `int inv_icm426xx_set_reg_bank` (struct `inv_icm426xx` \*s, `uint8_t` bank)  
*Set register bank index.*
- `int inv_icm426xx_init` (struct `inv_icm426xx` \*s, struct `inv_icm426xx_serif` \*serif, void(\*sensor\_event\_cb)(`inv_icm426xx_sensor_event_t` \*event))  
*Configure the serial interface used to access the device and execute hardware initialization.*
- `int inv_icm426xx_device_reset` (struct `inv_icm426xx` \*s)  
*Perform a soft reset of the device.*
- `int inv_icm426xx_get_who_am_i` (struct `inv_icm426xx` \*s, `uint8_t` \*who\_am\_i)  
*return WHOAMI value*
- `int inv_icm426xx_force_clock_source` (struct `inv_icm426xx` \*s, `ICM426XX_INTF_CONFIG1_ACCEL_LP_CLK_t` clk\_src)  
*Configure Accel clock source.*
- `int inv_icm426xx_enable_accel_low_power_mode` (struct `inv_icm426xx` \*s)  
*Enable/put accel in low power mode.*
- `int inv_icm426xx_enable_accel_low_noise_mode` (struct `inv_icm426xx` \*s)  
*Enable/put accel in low noise mode.*
- `int inv_icm426xx_disable_accel` (struct `inv_icm426xx` \*s)  
*Disable all 3 axes of accel.*
- `int inv_icm426xx_enable_gyro_low_noise_mode` (struct `inv_icm426xx` \*s)  
*Enable/put gyro in low noise mode.*
- `int inv_icm426xx_disable_gyro` (struct `inv_icm426xx` \*s)  
*Disable all 3 axes of gyro.*
- `int inv_icm426xx_enable_fsyc` (struct `inv_icm426xx` \*s)  
*Enable fsync tagging fonctionnality.*
- `int inv_icm426xx_disable_fsyc` (struct `inv_icm426xx` \*s)  
*Disable fsync tagging fonctionnality.*
- `int inv_icm426xx_configure_timestamp_resolution` (struct `inv_icm426xx` \*s, `ICM426XX_TMST_CONFIG_RESOLUTION_t` resol)  
*Configure timestamp resolution from FIFO.*
- `int inv_icm426xx_set_config_ibi` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Configure which interrupt source can trigger ibi interruptions.*
- `int inv_icm426xx_get_config_ibi` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Retrieve interrupts configuration.*
- `int inv_icm426xx_set_config_int1` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Configure which interrupt source can trigger INT1.*
- `int inv_icm426xx_get_config_int1` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Retrieve interrupts configuration.*
- `int inv_icm426xx_set_config_int2` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Configure which interrupt source can trigger INT2.*
- `int inv_icm426xx_get_config_int2` (struct `inv_icm426xx` \*s, `inv_icm426xx_interrupt_parameter_t` \*interrupt\_to\_configure)  
*Retrieve interrupts configuration.*
- `int inv_icm426xx_get_data_from_registers` (struct `inv_icm426xx` \*s)



- Read all registers containing data (temperature, accelerometer and gyroscope).*

  - `int inv_icm426xx_get_data_from_fifo` (struct `inv_icm426xx` \*s)

*Read all available packets from the FIFO.*
- `uint32_t inv_icm426xx_convert_odr_bitfield_to_us` (uint32\_t odr\_bitfield)

*Converts ICM426XX\_ACCEL\_CONFIG0\_ODR\_t or ICM426XX\_GYRO\_CONFIG0\_ODR\_t enums to period expressed in us.*
- `int inv_icm426xx_set_accel_frequency` (struct `inv_icm426xx` \*s, const `ICM426XX_ACCEL_CONFIG0_ODR_t` frequency)

*Configure accel Output Data Rate.*
- `int inv_icm426xx_set_gyro_frequency` (struct `inv_icm426xx` \*s, const `ICM426XX_GYRO_CONFIG0_ODR_t` frequency)

*Configure gyro Output Data Rate.*
- `int inv_icm426xx_set_accel_fsr` (struct `inv_icm426xx` \*s, `ICM426XX_ACCEL_CONFIG0_FS_SEL_t` accel\_fsr\_g)

*Set accel full scale range.*
- `int inv_icm426xx_get_accel_fsr` (struct `inv_icm426xx` \*s, `ICM426XX_ACCEL_CONFIG0_FS_SEL_t` \*accel\_fsr\_g)

*Access accel full scale range.*
- `int inv_icm426xx_set_gyro_fsr` (struct `inv_icm426xx` \*s, `ICM426XX_GYRO_CONFIG0_FS_SEL_t` gyro\_fsr\_dps)

*Set gyro full scale range.*
- `int inv_icm426xx_get_gyro_fsr` (struct `inv_icm426xx` \*s, `ICM426XX_GYRO_CONFIG0_FS_SEL_t` \*gyro\_fsr\_dps)

*Access gyro full scale range.*
- `int inv_icm426xx_set_accel_lp_avg` (struct `inv_icm426xx` \*s, `ICM426XX_GYRO_ACCEL_CONFIG0_ACCEL_FILT_AVG_t` acc\_avg)

*Set accel Low-Power averaging value.*
- `int inv_icm426xx_set_accel_ln_bw` (struct `inv_icm426xx` \*s, `ICM426XX_GYRO_ACCEL_CONFIG0_ACCEL_FILT_BW_t` acc\_bw)

*Set accel Low-Noise bandwidth value.*
- `int inv_icm426xx_set_gyro_ln_bw` (struct `inv_icm426xx` \*s, `ICM426XX_GYRO_ACCEL_CONFIG0_GYRO_FILT_BW_t` gyr\_bw)

*Set gyro Low-Noise bandwidth value.*
- `int inv_icm426xx_reset_fifo` (struct `inv_icm426xx` \*s)

*reset ICM426XX fifo*
- `int inv_icm426xx_enable_timestamp_to_register` (struct `inv_icm426xx` \*s)

*Enable the 20bits-timestamp register access to read in a reliable way the strobed timestamp.*
- `int inv_icm426xx_disable_timestamp_to_register` (struct `inv_icm426xx` \*s)

*Disable the 20bits-timestamp register access.*
- `int inv_icm426xx_get_current_timestamp` (struct `inv_icm426xx` \*s, uint32\_t \*icm\_time)

*Get the timestamp value of icm426xx from register.*
- `int inv_icm426xx_enable_clkin_rtc` (struct `inv_icm426xx` \*s, uint8\_t enable)

*Enable or disable CLKIN/RTC capability.*
- `int inv_icm426xx_get_clkin_rtc_status` (struct `inv_icm426xx` \*s)

*Get CLKIN/RTC feature status.*
- `int inv_icm426xx_enable_high_resolution_fifo` (struct `inv_icm426xx` \*s)

*Enable 20 bits raw acc and raw gyr data in fifo.*
- `int inv_icm426xx_disable_high_resolution_fifo` (struct `inv_icm426xx` \*s)

*Disable 20 bits raw acc and raw gyr data in fifo.*
- `int inv_icm426xx_configure_fifo` (struct `inv_icm426xx` \*s, `INV_ICM426XX_FIFO_CONFIG_t` fifo\_config)

*Configure Fifo to select the way data are gathered.*
- `int inv_icm426xx_configure_fifo_wm` (struct `inv_icm426xx` \*s, uint16\_t wm)

*Configure Fifo watermark (also referred to as fifo threshold)*

- uint32\_t [inv\\_icm426xx\\_get\\_fifo\\_timestamp\\_resolution\\_us\\_q24](#) (struct [inv\\_icm426xx](#) \*s)

*Get FIFO timestamp resolution.*

- uint32\_t [inv\\_icm426xx\\_get\\_reg\\_timestamp\\_resolution\\_us\\_q24](#) (struct [inv\\_icm426xx](#) \*s)

*Get register timestamp resolution.*

- const char \* [inv\\_icm426xx\\_get\\_version](#) (void)

*Return driver version x.y.z-suffix as a char array.*

### 4.1.1 Detailed Description

High-level function to setup an Icm426xx device.

### 4.1.2 Macro Definition Documentation

#### 4.1.2.1 #define ICM426XX\_FIFO\_MIRRORING\_SIZE 16 \* 129

Icm426xx maximum buffer size mirrored from FIFO at polling time.

#### Warning

fifo\_idx type variable must be large enough to parse the FIFO\_MIRRORING\_SIZE

#### 4.1.2.2 #define INV\_ICM426XX\_LIGHTWEIGHT\_DRIVER 0

Lighthen driver logic by stripping out procedures on transitions.

In the nominal case, ie. when sensors are enabled and their output has settled, ICM-426XX will not need the logic to handle each transition. They are part of each API function so this code will be linked in regardless. This might not be desirable for the most size-constrained platforms and it can be avoided by setting this define to 1.

### 4.1.3 Enumeration Type Documentation

#### 4.1.3.1 enum INV\_ICM426XX\_FIFO\_CONFIG\_t

Configure Fifo usage.

#### Enumerator

**INV\_ICM426XX\_FIFO\_DISABLED** Fifo is disabled and data source is sensors registers.

**INV\_ICM426XX\_FIFO\_ENABLED** Fifo is used as data source.

## 4.1.3.2 enum inv\_icm426xx\_sensor

Sensor identifier for UI control and OIS function.

Enumerator

**INV\_ICM426XX\_SENSOR\_ACCEL** Accelerometer (UI control path)  
**INV\_ICM426XX\_SENSOR\_GYRO** Gyroscope (UI control path)  
**INV\_ICM426XX\_SENSOR\_FSYNC\_EVENT** Used by OIS and UI control layers.  
**INV\_ICM426XX\_SENSOR\_OIS** Only used by OIS layer.  
**INV\_ICM426XX\_SENSOR\_TEMPERATURE** Chip temperature, enabled by default. However, it will be reported only if Accel and/or Gyro are also enabled. The Temperature's ODR (Output Data Rate) will match the ODR of Accel or Gyro, or the fastest if both are enabled  
**INV\_ICM426XX\_SENSOR\_TAP** Tap and Double tap.  
**INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER\_EVENT** Pedometer: step is detected.  
**INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER\_COUNT** Pedometer: step counter.  
**INV\_ICM426XX\_SENSOR\_DMP\_TILT** Tilt.  
**INV\_ICM426XX\_SENSOR\_DMP\_R2W** Raise to wake.

## 4.1.4 Function Documentation

## 4.1.4.1 int inv\_icm426xx\_configure\_fifo ( struct inv\_icm426xx \* s, INV\_ICM426XX\_FIFO\_CONFIG\_t fifo\_config )

Configure Fifo to select the way data are gathered.

Parameters

in	<i>fifo_config</i>	Fifo configuration method : if enabled data are coming from fifo and Interrupt is configured on Fifo Watermark if disabled data are coming from sensor registers and Interrupt is configured on Data ready
----	--------------------	--

See also

[INV\\_ICM426XX\\_FIFO\\_CONFIG\\_t](#)

## 4.1.4.2 int inv\_icm426xx\_configure\_fifo\_wm ( struct inv\_icm426xx \* s, uint16\_t wm )

Configure Fifo watermark (also referred to as fifo threshold)

Parameters

in	<i>wm</i>	Watermark value
----	-----------	-----------------

4.1.4.3 `int inv_icm426xx_configure_timestamp_resolution ( struct inv_icm426xx * s, ICM426XX_TMST_CONFIG_RESOL_t resol )`

Configure timestamp resolution from FIFO.

#### Parameters

<code>in</code>	<code>resol</code>	The expected resolution of the timestamp. See enum ICM426XX_TMST_CONFIG_RESOL_t.
-----------------	--------------------	--

#### Returns

0 on success, negative value on error.

#### Warning

The resolution will have no effect if RTC is enabled

4.1.4.4 `uint32_t inv_icm426xx_convert_odr_bitfield_to_us ( uint32_t odr_bitfield )`

Converts ICM426XX\_ACCEL\_CONFIG0\_ODR\_t or ICM426XX\_GYRO\_CONFIG0\_ODR\_t enums to period expressed in us.

#### Parameters

<code>in</code>	<code>odr_bitfield</code>	An ICM426XX_ACCEL_CONFIG0_ODR_t or ICM426XX_GYRO_CONFIG0_ODR_t enum
-----------------	---------------------------	---

#### Returns

The corresponding period expressed in us

4.1.4.5 `int inv_icm426xx_device_reset ( struct inv_icm426xx * s )`

Perform a soft reset of the device.

#### Returns

0 on success, negative value on error.

4.1.4.6 `int inv_icm426xx_disable_accel ( struct inv_icm426xx * s )`

Disable all 3 axes of accel.

#### Returns

0 on success, negative value on error.

If both accel and gyro are turned off as a result of this function, they will also be removed from the FIFO and a FIFO reset will be performed (to guarantee no side effects until the next enable sensor call)

#### Warning

`inv_icm426xx::register_cache::pwr_mngt_0_reg` is modified by this function

#### 4.1.4.7 int inv\_icm426xx\_disable\_fsync ( struct inv\_icm426xx \* s )

Disable fsync tagging fonctionnality.

In details it:

- disables fsync
- disables timestamp to registers. Once fsync is disabled timestamp is pushed to fifo instead of fsync counter. So in order to decrease power consumption, timestamp is no more available in registers.
- disables fsync related interrupt

##### Returns

0 on success, negative value on error.

#### 4.1.4.8 int inv\_icm426xx\_disable\_gyro ( struct inv\_icm426xx \* s )

Disable all 3 axes of gyro.

##### Returns

0 on success, negative value on error.

If both accel and gyro are turned off as a result of this function, they will also be removed from the FIFO and a FIFO reset will be performed (to guarantee no side effects until the next enable sensor call)

##### Warning

inv\_icm426xx::register\_cache::pwr\_mngt\_0\_reg is modified by this function

#### 4.1.4.9 int inv\_icm426xx\_disable\_high\_resolution\_fifo ( struct inv\_icm426xx \* s )

Disable 20 bits raw acc and raw gyr data in fifo.

##### Returns

0 on success, negative return code otherwise

#### 4.1.4.10 int inv\_icm426xx\_disable\_timestamp\_to\_register ( struct inv\_icm426xx \* s )

Disable the 20bits-timestamp register access.

Register read always return 0's.

##### Returns

0 on success, negative value on error.

#### 4.1.4.11 `int inv_icm426xx_enable_accel_low_noise_mode ( struct inv_icm426xx * s )`

Enable/put accel in low noise mode.

##### Returns

0 on success, negative value on error.

It enables accel and gyro data in the FIFO (so the packet format is 16 bytes). If called first, the configuration will be applied, otherwise it will be ignored if the FIFO is not empty (but since the new configuration is identical it is not a issue).

##### Warning

`inv_icm426xx::register_cache::pwr_mngt_0_reg` is modified by this function

#### 4.1.4.12 `int inv_icm426xx_enable_accel_low_power_mode ( struct inv_icm426xx * s )`

Enable/put accel in low power mode.

##### Returns

0 on success, negative value on error.

It enables accel and gyro data in the FIFO (so the packet format is 16 bytes). If called first, the configuration will be applied, otherwise it will be ignored if the FIFO is not empty (but since the new configuration is identical it is not a issue).

##### Warning

`inv_icm426xx::register_cache::pwr_mngt_0_reg` is modified by this function

#### 4.1.4.13 `int inv_icm426xx_enable_clkin_rtc ( struct inv_icm426xx * s, uint8_t enable )`

Enable or disable CLKIN/RTC capability.

##### Parameters

<code>in</code>	<code>enable</code>	1 if external 32kHz is provided to ICM, 0 otherwise
-----------------	---------------------	---

##### Returns

0 on success, negative value on error.

##### Warning

In case CLKIN is disabled, it is recommended to call [inv\\_icm426xx\\_configure\\_timestamp\\_resolution\(\)](#) just afterwards so that timestamp resolution is in line with system request  
`inv_icm426xx::register_cache::intf_cfg_1_reg` is modified by this function

#### 4.1.4.14 `int inv_icm426xx_enable_fsync ( struct inv_icm426xx * s )`

Enable fsync tagging fonctionnality.

In details it:

- enables fsync
- enables timestamp to registers. Once fsync is enabled fsync counter is pushed to fifo instead of timestamp. So timestamp is made available in registers. Note that this increase power consumption.
- enables fsync related interrupt

##### Returns

0 on success, negative value on error.

#### 4.1.4.15 `int inv_icm426xx_enable_gyro_low_noise_mode ( struct inv_icm426xx * s )`

Enable/put gyro in low noise mode.

##### Returns

0 on success, negative value on error.

It enables gyro and accel data in the FIFO (so the packet format is 16 bytes). If called first, the configuration will be applied, otherwise it will be ignored if the FIFO is not empty (but since the new configuration is identical it is not a issue).

##### Warning

`inv_icm426xx::register_cache::pwr_mngt_0_reg` is modified by this function

#### 4.1.4.16 `int inv_icm426xx_enable_high_resolution_fifo ( struct inv_icm426xx * s )`

Enable 20 bits raw acc and raw gyr data in fifo.

##### Returns

0 on success, negative return code otherwise

#### 4.1.4.17 `int inv_icm426xx_enable_timestamp_to_register ( struct inv_icm426xx * s )`

Enable the 20bits-timestamp register access to read in a reliable way the strobed timestamp.

To do that, the fine clock is forced enabled at some power cost.

##### Returns

0 on success, negative value on error.

#### 4.1.4.18 `int inv_icm426xx_force_clock_source ( struct inv_icm426xx * s, ICM426XX_INTF_CONFIG1_ACCEL_LP_CLK_t clk_src )`

Configure Accel clock source.

**Parameters**

in	<i>new</i>	clock source to use
----	------------	---------------------

**Returns**

0 on success, negative value on error

**Note**

Transitions when enabling/disabling sensors are already handled by the driver. This function is here only to force a specific clock source and shall be used with care.

4.1.4.19 `int inv_icm426xx_get_accel_fsr ( struct inv_icm426xx * s, ICM426XX_ACCEL_CONFIG0_FS_SEL_t * accel_fsr_g )`

Access accel full scale range.

**Parameters**

out	<i>accel_fsr_g</i>	current full scale range.
-----	--------------------	---------------------------

**See also**

[ICM426XX\\_ACCEL\\_CONFIG0\\_FS\\_SEL\\_t](#).

**Returns**

0 on success, negative value on error.

**Warning**

`inv_icm426xx::register_cache::accel_cfg_0_reg` is relied upon by this function

4.1.4.20 `int inv_icm426xx_get_clkin_rtc_status ( struct inv_icm426xx * s )`

Get CLKIN/RTC feature status.

**Returns**

0 if CLKIN is disabled, 1 if enabled.

**Warning**

In case CLKIN is disabled, it is recommended to call [inv\\_icm426xx\\_configure\\_timestamp\\_resolution\(\)](#) just afterwards so that timestamp resolution is in line with system request  
`inv_icm426xx::register_cache::intf_cfg_1_reg` is relied upon by this function

4.1.4.21 `int inv_icm426xx_get_config_ibi ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t * interrupt_to_configure )`

Retrieve interrupts configuration.



## Parameters

in	<i>interrupt_to_configure</i>	structure with the corresponding state to manage IBI interruptions.
----	-------------------------------	---

## Returns

0 on success, negative value on error.

4.1.4.22 `int inv_icm426xx_get_config_int1 ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t * interrupt_to_configure )`

Retrieve interrupts configuration.

## Parameters

in	<i>interrupt_to_configure</i>	structure with the corresponding state to manage INT1.
----	-------------------------------	--

## Returns

0 on success, negative value on error.

4.1.4.23 `int inv_icm426xx_get_config_int2 ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t * interrupt_to_configure )`

Retrieve interrupts configuration.

## Parameters

in	<i>interrupt_to_configure</i>	structure with the corresponding state to manage INT2.
----	-------------------------------	--

## Returns

0 on success, negative value on error.

4.1.4.24 `int inv_icm426xx_get_current_timestamp ( struct inv_icm426xx * s, uint32_t * icm_time )`

Get the timestamp value of icm426xx from register.

## Parameters

in	<i>icm_time</i>	timestamp read from register
----	-----------------	------------------------------

**Returns**

0 on success, negative value on error.

**Warning**

Prior to call this API, the read access to timestamp register must be enabled (see [inv\\_icm426xx\\_enable\\_timestamp\\_to\\_register\(\)](#) function)

**4.1.4.25 int inv\_icm426xx\_get\_data\_from\_fifo ( struct inv\_icm426xx \* s )**

Read all available packets from the FIFO.

For each packet function builds a sensor event containing packet data and validity information. Then it calls sensor\_event\_cb function passed in parameter of inv\_icm426xx\_init function for each packet.

**Returns**

0 on success, negative value on error.

**4.1.4.26 int inv\_icm426xx\_get\_data\_from\_registers ( struct inv\_icm426xx \* s )**

Read all registers containing data (temperature, accelerometer and gyroscope).

Then it calls sensor\_event\_cb function passed in parameter of inv\_icm426xx\_init function for each packet

**Returns**

0 on success, negative value on error.

**4.1.4.27 uint32\_t inv\_icm426xx\_get\_fifo\_timestamp\_resolution\_us\_q24 ( struct inv\_icm426xx \* s )**

Get FIFO timestamp resolution.

**Returns**

the timestamp resolution in us as a q24 or 0 in case of error

**4.1.4.28 int inv\_icm426xx\_get\_gyro\_fsr ( struct inv\_icm426xx \* s, ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_t \* gyro\_fsr\_dps )**

Access gyro full scale range.

**Parameters**

out	<i>gyro_fsr_dps</i>	current full scale range.
-----	---------------------	---------------------------

See also

[ICM426XX\\_GYRO\\_CONFIG0\\_FS\\_SEL\\_t](#).

Returns

0 on success, negative value on error.

Warning

inv\_icm426xx::register\_cache::gyro\_cfg\_0\_reg is relied upon by this function

4.1.4.29 `uint32_t inv_icm426xx_get_reg_timestamp_resolution_us_q24 ( struct inv_icm426xx * s )`

Get register timestamp resolution.

Returns

the timestamp resolution in us as a q24 or 0 in case of error

4.1.4.30 `const char* inv_icm426xx_get_version ( void )`

Return driver version x.y.z-suffix as a char array.

Return values

<i>driver</i>	version a char array "x.y.z-suffix"
---------------	-------------------------------------

4.1.4.31 `int inv_icm426xx_get_who_am_i ( struct inv_icm426xx * s, uint8_t * who_am_i )`

return WHOAMI value

Parameters

out	<i>who_↔ am_i</i>	WHOAMI for device
-----	-----------------------	-------------------

Returns

0 on success, negative value on error

4.1.4.32 `int inv_icm426xx_init ( struct inv_icm426xx * s, struct inv_icm426xx_serif * serif,  
void (*)(inv_icm426xx_sensor_event_t *event) sensor_event_cb )`

Configure the serial interface used to access the device and execute hardware initialization.

This functions first configures serial interface passed in parameter to make sure device is accessible both in read and write. Thus no serial access should be done before succesfully executing the present function.

Then if requested serial interface is a primary interface (aka UI interface or AP interface), this function initializes the device using the following hardware settings:

- gyroscope fsr = 2000dps
- accelerometer fsr = 4g
- set timestamp resolution to 16us
- enable FIFO mechanism with the following configuration:
  - FIFO record mode i.e FIFO count unit is packet
  - FIFO snapshot mode i.e drop the data when the FIFO overflows
  - Timestamp is logged in FIFO
  - Little Endian fifo\_count and fifo\_data
  - generate FIFO threshold interrupt when packet count reaches FIFO watermark
  - set FIFO watermark to 1 packet
  - enable temperature and timestamp data to go to FIFO

In case requested serial interface is an auxliary interface (i.e. AUX1 or AUX2) this function returns an error.

#### Parameters

in	<i>s</i>	driver structure. Note that first field of this structure MUST be a struct <a href="#">inv_icm426xx_serif</a> .
in	<i>serif</i>	pointer on serial interface structure to be used to access icm426xx.
in	<i>sensor_event_cb</i>	callback executed by <code>inv_icm426xx_get_data_from_fifo</code> function each time it extracts some valid data from fifo. Or <code>inv_icm426xx_get_data_from_registers</code> read data from register. Thus this parameter is optional as long as <code>inv_icm426xx_get_data_from_fifo/inv_icm426xx_get_data_from_registers</code> function is not used.

#### Returns

0 on success, negative value on error.

#### 4.1.4.33 int inv\_icm426xx\_reset\_fifo ( struct inv\_icm426xx \* s )

reset ICM426XX fifo

#### Returns

0 on success, negative value on error.

#### 4.1.4.34 int inv\_icm426xx\_set\_accel\_frequency ( struct inv\_icm426xx \* s, const ICM426XX\_ACCEL\_CONFIG0\_OD↔ R\_t frequency )

Configure accel Output Data Rate.

## Parameters

in	<i>frequency</i>	The requested frequency.
----	------------------	--------------------------

## See also

[ICM426XX\\_ACCEL\\_CONFIG0\\_ODR\\_t](#)

## Returns

0 on success, negative value on error.

## Warning

inv\_icm426xx::register\_cache::accel\_cfg\_0\_reg is modified by this function

4.1.4.35 int inv\_icm426xx\_set\_accel\_fsr ( struct inv\_icm426xx \* s, ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_t accel\_fsr\_g )

Set accel full scale range.

## Parameters

in	<i>accel_fsr↔ _g</i>	requested full scale range.
----	--------------------------	-----------------------------

## See also

[ICM426XX\\_ACCEL\\_CONFIG0\\_FS\\_SEL\\_t](#).

## Returns

0 on success, negative value on error.

## Warning

inv\_icm426xx::register\_cache::accel\_cfg\_0\_reg is modified by this function

4.1.4.36 int inv\_icm426xx\_set\_accel\_ln\_bw ( struct inv\_icm426xx \* s, ICM426XX\_GYRO\_ACCEL\_CONFIG0\_ACCEL\_FILT\_↔BW\_t acc\_bw )

Set accel Low-Noise bandwidth value.

## Parameters

in	<i>acc_bw</i>	requested averaging value
----	---------------	---------------------------

See also

ICM426XX\_GYRO\_ACCEL\_CONFIG0\_ACCEL\_FILT\_BW\_t

Returns

0 on success, negative value on error.

4.1.4.37 `int inv_icm426xx_set_accel_lp_avg ( struct inv_icm426xx * s, ICM426XX_GYRO_ACCEL_CONFIG0_ACCEL_FILT_A↵  
_AVG_t acc_avg )`

Set accel Low-Power averaging value.

Parameters

in	<i>acc_avg</i>	requested averaging value
----	----------------	---------------------------

See also

ICM426XX\_GYRO\_ACCEL\_CONFIG0\_ACCEL\_FILT\_AVG\_t

Returns

0 on success, negative value on error.

4.1.4.38 `int inv_icm426xx_set_config_ibi ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t *  
interrupt_to_configure )`

Configure which interrupt source can trigger ibi interruptions.

Parameters

in	<i>interrupt_to_configure</i>	structure with the corresponding state to manage ibi interruptions.
----	-------------------------------	---

Returns

0 on success, negative value on error.

4.1.4.39 `int inv_icm426xx_set_config_int1 ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t *  
interrupt_to_configure )`

Configure which interrupt source can trigger INT1.

Parameters

in	<i>interrupt_to_configure</i>	structure with the corresponding state to manage INT1.
----	-------------------------------	--

**Returns**

0 on success, negative value on error.

4.1.4.40 `int inv_icm426xx_set_config_int2 ( struct inv_icm426xx * s, inv_icm426xx_interrupt_parameter_t * interrupt_to_configure )`

Configure which interrupt source can trigger INT2.

**Parameters**

in	<i>interrupt_to_configure</i>	structure with the corresponding state to INT2.
----	-------------------------------	---

**Returns**

0 on success, negative value on error.

4.1.4.41 `int inv_icm426xx_set_gyro_frequency ( struct inv_icm426xx * s, const ICM426XX_GYRO_CONFIG0_ODR_t frequency )`

Configure gyro Output Data Rate.

**Parameters**

in	<i>frequency</i>	The requested frequency.
----	------------------	--------------------------

**See also**

[ICM426XX\\_GYRO\\_CONFIG0\\_ODR\\_t](#)

**Returns**

0 on success, negative value on error.

**Warning**

inv\_icm426xx::register\_cache::gyro\_cfg\_0\_reg is modified by this function

4.1.4.42 `int inv_icm426xx_set_gyro_fsr ( struct inv_icm426xx * s, ICM426XX_GYRO_CONFIG0_FS_SEL_t gyro_fsr_dps )`

Set gyro full scale range.

**Parameters**

in	<i>gyro_fsr_dps</i>	requested full scale range.
----	---------------------	-----------------------------

See also

[ICM426XX\\_GYRO\\_CONFIG0\\_FS\\_SEL\\_t](#).

Returns

0 on success, negative value on error.

Warning

inv\_icm426xx::register\_cache::gyro\_cfg\_0\_reg is modified by this function

4.1.4.43 `int inv_icm426xx_set_gyro_ln_bw ( struct inv_icm426xx * s, ICM426XX_GYRO_ACCEL_CONFIG0_GYRO_FILT_BW_t gyr_bw )`

Set gyro Low-Noise bandwidth value.

Parameters

in	<i>gyr_bw</i>	requested averaging value
----	---------------	---------------------------

See also

[ICM426XX\\_GYRO\\_ACCEL\\_CONFIG0\\_GYRO\\_FILT\\_BW\\_t](#)

Returns

0 on success, negative value on error.

4.1.4.44 `int inv_icm426xx_set_reg_bank ( struct inv_icm426xx * s, uint8_t bank )`

Set register bank index.

Parameters

<i>bank</i>	new bank to be set
-------------	--------------------

Returns

0 on success, negative value otherwise



## 4.2 Icm426xx driver high level functions related to APEX

High-level function to setup an Icm426xx device.

### Files

- file [Icm426xxDriver\\_HL\\_apex.h](#)  
*High-level function to setup an Icm426xx device.*

### Classes

- struct [inv\\_icm426xx\\_tap\\_parameters\\_t](#)  
*Icm426xx TAP inputs parameters definition.*
- struct [inv\\_icm426xx\\_apex\\_parameters](#)  
*Icm426xx APEX inputs parameters definition.*
- struct [inv\\_icm426xx\\_apex\\_step\\_activity](#)  
*APEX pedometer outputs.*
- struct [inv\\_icm426xx\\_tap\\_data](#)  
*TAP outputs.*

### Typedefs

- typedef struct [inv\\_icm426xx\\_apex\\_parameters](#) [inv\\_icm426xx\\_apex\\_parameters\\_t](#)  
*Icm426xx APEX inputs parameters definition.*
- typedef struct [inv\\_icm426xx\\_apex\\_step\\_activity](#) [inv\\_icm426xx\\_apex\\_step\\_activity\\_t](#)  
*APEX pedometer outputs.*
- typedef struct [inv\\_icm426xx\\_tap\\_data](#) [inv\\_icm426xx\\_tap\\_data\\_t](#)  
*TAP outputs.*

### Functions

- int [inv\\_icm426xx\\_configure\\_smd\\_wom](#) (struct [inv\\_icm426xx](#) \*s, const uint8\_t x\_th, const uint8\_t y\_th, const uint8\_t z\_th, ICM426XX\_SMD\_CONFIG\_WOM\_INT\_MODE\_t wom\_int, ICM426XX\_SMD\_CONFIG\_WOM\_MODE\_t wom\_mode)  
*Configure Wake On Motion and SMD thresholds.*
- int [inv\\_icm426xx\\_enable\\_wom](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable Wake On Motion.*
- int [inv\\_icm426xx\\_disable\\_wom](#) (struct [inv\\_icm426xx](#) \*s)  
*Disable Wake On Motion.*
- int [inv\\_icm426xx\\_enable\\_smd](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable Significant Motion Detection.*
- int [inv\\_icm426xx\\_disable\\_smd](#) (struct [inv\\_icm426xx](#) \*s)  
*Disable Significant Motion Detection.*
- int [inv\\_icm426xx\\_init\\_tap\\_parameters\\_struct](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_tap\\_parameters\\_t](#) \*tap\_inputs)  
*Fill the TAP parameters structure with all the default parameters for TAP algorithm.*
- int [inv\\_icm426xx\\_configure\\_tap\\_parameters](#) (struct [inv\\_icm426xx](#) \*s, const [inv\\_icm426xx\\_tap\\_parameters\\_t](#) \*tap\_inputs)

- Configure TAP.*

  - int `inv_icm426xx_get_tap_parameters` (struct `inv_icm426xx` \*s, `inv_icm426xx_tap_parameters_t` \*tap\_params)
 

*Returns current TAP parameters.*
  - int `inv_icm426xx_enable_tap` (struct `inv_icm426xx` \*s)
 

*Enable TAP.*
  - int `inv_icm426xx_disable_tap` (struct `inv_icm426xx` \*s)
 

*Disable TAP.*
  - int `inv_icm426xx_init_apex_parameters_struct` (struct `inv_icm426xx` \*s, `inv_icm426xx_apex_parameters_t` \*apex\_inputs)
 

*Fill the APEX parameters structure with all the default parameters for APEX algorithms (pedometer, tilt)*
  - int `inv_icm426xx_configure_apex_parameters` (struct `inv_icm426xx` \*s, const `inv_icm426xx_apex_parameters_t` \*apex\_inputs)
 

*Configures DMP parameters for APEX algorithms (pedometer, tilt).*
  - int `inv_icm426xx_get_apex_parameters` (struct `inv_icm426xx` \*s, `inv_icm426xx_apex_parameters_t` \*apex\_params)
 

*Returns current DMP parameters for APEX algorithms (pedometer, tilt).*
  - int `inv_icm426xx_set_apex_frequency` (struct `inv_icm426xx` \*s, const `ICM426XX_APEX_CONFIG0_DMP_ODR_t` frequency)
 

*Configure DMP Output Data Rate for APEX algorithms (pedometer, tilt)*
  - int `inv_icm426xx_start_dmp` (struct `inv_icm426xx` \*s)
 

*Start DMP for APEX algorithms.*
  - int `inv_icm426xx_enable_apex_pedometer` (struct `inv_icm426xx` \*s)
 

*Enable APEX algorithm Pedometer.*
  - int `inv_icm426xx_disable_apex_pedometer` (struct `inv_icm426xx` \*s)
 

*Disable APEX algorithm Pedometer.*
  - int `inv_icm426xx_enable_apex_r2w` (struct `inv_icm426xx` \*s)
 

*Enable APEX algorithm Raise to wake.*
  - int `inv_icm426xx_disable_apex_r2w` (struct `inv_icm426xx` \*s)
 

*Disable APEX algorithm Raise to wake.*
  - int `inv_icm426xx_enable_apex_tilt` (struct `inv_icm426xx` \*s)
 

*Enable APEX algorithm Tilt.*
  - int `inv_icm426xx_disable_apex_tilt` (struct `inv_icm426xx` \*s)
 

*Disable APEX algorithm Tilt.*
  - int `inv_icm426xx_get_apex_data_activity` (struct `inv_icm426xx` \*s, `inv_icm426xx_apex_step_activity_t` \*apex\_activity)
 

*Retrieve APEX pedometer outputs and format them.*
  - int `inv_icm426xx_get_tap_data` (struct `inv_icm426xx` \*s, `inv_icm426xx_tap_data_t` \*tap\_data)
 

*Retrieve tap outputs.*

#### 4.2.1 Detailed Description

High-level function to setup an ICM426xx device.

#### 4.2.2 Function Documentation

##### 4.2.2.1 int inv\_icm426xx\_configure\_apex\_parameters ( struct inv\_icm426xx \* s, const inv\_icm426xx\_apex\_parameters\_t \* apex\_inputs )

Configures DMP parameters for APEX algorithms (pedometer, tilt).

This programmable parameters will be decoded and propagate to the SRAM to be executed at DMP start.

## Parameters

in	<i>apex_inputs</i>	The requested input parameters. See
----	--------------------	-------------------------------------

## See also

[inv\\_icm426xx\\_apex\\_parameters\\_t](#)

## Warning

APEX inputs can't change on the fly, this API should be called before enabling any APEX features.  
 APEX configuration can't be done too frequently, but only once every 10ms. Otherwise it can create unknown behavior.

## Returns

0 on success, negative value on error.

**4.2.2.2** `int inv_icm426xx_configure_smd_wom ( struct inv_icm426xx * s, const uint8_t x_th, const uint8_t y_th, const uint8_t z_th, ICM426XX_SMD_CONFIG_WOM_INT_MODE_t wom_int, ICM426XX_SMD_CONFIG_WOM_MODE_t wom_mode )`

Configure Wake On Motion and SMD thresholds.

## Parameters

in	<i>x_th</i>	threshold value for the Wake on Motion Interrupt for X-axis accelerometer.
in	<i>y_th</i>	threshold value for the Wake on Motion Interrupt for Y-axis accelerometer.
in	<i>z_th</i>	threshold value for the Wake on Motion Interrupt for Z-axis accelerometer.
in	<i>wom_int</i>	select which mode between AND/OR is used to generate interrupt.
in	<i>wom_mode</i>	select which comparison mode is used for WoM detection.

## Returns

0 on success, negative value on error.

**4.2.2.3** `int inv_icm426xx_configure_tap_parameters ( struct inv_icm426xx * s, const inv_icm426xx_tap_parameters_t * tap_inputs )`

Configure TAP.

## Parameters

in	<i>tap_inputs</i>	The requested input parameters. See
----	-------------------	-------------------------------------

See also

[inv\\_icm426xx\\_tap\\_parameters\\_t](#)

Returns

0 on success, negative value on error.

4.2.2.4 int inv\_icm426xx\_disable\_apex\_pedometer ( struct inv\_icm426xx \* s )

Disable APEX algorithm Pedometer.

Returns

0 on success, negative value on error.

4.2.2.5 int inv\_icm426xx\_disable\_apex\_r2w ( struct inv\_icm426xx \* s )

Disable APEX algorithm Raise to wake.

Returns

0 on success, negative value on error.

4.2.2.6 int inv\_icm426xx\_disable\_apex\_tilt ( struct inv\_icm426xx \* s )

Disable APEX algorithm Tilt.

Returns

0 on success, negative value on error.

4.2.2.7 int inv\_icm426xx\_disable\_smd ( struct inv\_icm426xx \* s )

Disable Significant Motion Detection.

Disables SMD event generation and disables SMD interrupt.

Returns

0 on success, negative value on error.

**4.2.2.8 int inv\_icm426xx\_disable\_tap ( struct inv\_icm426xx \* s )**

Disable TAP.

**Returns**

0 on success, negative value on error.

**4.2.2.9 int inv\_icm426xx\_disable\_wom ( struct inv\_icm426xx \* s )**

Disable Wake On Motion.

Disables WoM event generation and reconfigures interrupt to fire on Fifo water-mark.

**Returns**

0 on success, negative value on error.

**4.2.2.10 int inv\_icm426xx\_enable\_apex\_pedometer ( struct inv\_icm426xx \* s )**

Enable APEX algorithm Pedometer.

note : Pedometer request to have the accelerometer enabled to works with accelerometer frequency less than dmp frequency.

**Returns**

0 on success, negative value on error.

**Warning**

Pedometer must be turned OFF to reconfigure it

**4.2.2.11 int inv\_icm426xx\_enable\_apex\_r2w ( struct inv\_icm426xx \* s )**

Enable APEX algorithm Raise to wake.

note : Raise to wake request to have the accelerometer enabled to works with accelerometer frequency less than dmp frequency.

**Returns**

0 on success, negative value on error.

#### 4.2.2.12 int inv\_icm426xx\_enable\_apex\_tilt ( struct inv\_icm426xx \* s )

Enable APEX algorithm Tilt.

note : Tilt request to have the accelerometer enabled to works with accelerometer frequency less than dmp frequency.

##### Returns

0 on success, negative value on error.

#### 4.2.2.13 int inv\_icm426xx\_enable\_smd ( struct inv\_icm426xx \* s )

Enable Significant Motion Detection.

note : SMD requests to have the accelerometer enabled to work. Enables SMD event generation and configures interrupt to fire on SMD event. WoM event will also be generated. To have good performance, it's recommended to set accelerometer ODR (Output Data Rate) to 20ms and the accelerometer in Low Power Mode.

##### Returns

0 on success, negative value on error.

#### 4.2.2.14 int inv\_icm426xx\_enable\_tap ( struct inv\_icm426xx \* s )

Enable TAP.

note : TAP requests to have the accelerometer enabled to work. To have good performance, it's recommended to set accelerometer ODR (Output Data Rate) to 1ms and the accelerometer in Low Noise Mode.

##### Returns

0 on success, negative value on error.

#### 4.2.2.15 int inv\_icm426xx\_enable\_wom ( struct inv\_icm426xx \* s )

Enable Wake On Motion.

note : WoM requests to have the accelerometer enabled to work. Enables WoM event generation and configures interrupt to fire on WoM event. As a consequence Fifo water-mark interrupt is disabled. To have good performance, it's recommended to set accelerometer ODR (Output Data Rate) to 20ms and the accelerometer in Low Power Mode.

##### Returns

0 on success, negative value on error.

#### 4.2.2.16 int inv\_icm426xx\_get\_apex\_data\_activity ( struct inv\_icm426xx \* s, inv\_icm426xx\_apex\_step\_activity\_t \* apex\_activity )

Retrieve APEX pedometer outputs and format them.

## Parameters

out	<i>apex_activity</i>	Apex step and activity data value.
-----	----------------------	------------------------------------

## Returns

0 in case of success, negative value on error. See enum `inv_error`

**4.2.2.17** `int inv_icm426xx_get_apex_parameters ( struct inv_icm426xx * s, inv_icm426xx_apex_parameters_t * apex_params )`

Returns current DMP parameters for APEX algorithms (pedometer, tilt).

## Parameters

out	<i>apex_params</i>	The current parameter, fetched from registers. See
-----	--------------------	--

## See also

[inv\\_icm426xx\\_apex\\_parameters\\_t](#)

## Returns

0 on success, negative value on error.

**4.2.2.18** `int inv_icm426xx_get_tap_data ( struct inv_icm426xx * s, inv_icm426xx_tap_data_t * tap_data )`

Retrieve tap outputs.

## Parameters

out	<i>tap_data</i>	Tap axis, direction and type
-----	-----------------	------------------------------

## Returns

0 in case of success, negative value on error. See enum `inv_error`

**4.2.2.19** `int inv_icm426xx_get_tap_parameters ( struct inv_icm426xx * s, inv_icm426xx_tap_parameters_t * tap_params )`

Returns current TAP parameters.

## Parameters

out	<i>tap_params</i>	The current parameter, fetched from registers. See
-----	-------------------	--

See also

[inv\\_icm426xx\\_tap\\_parameters\\_t](#)

Returns

0 on success, negative value on error.

4.2.2.20 `int inv_icm426xx_init_apex_parameters_struct ( struct inv_icm426xx * s, inv_icm426xx_apex_parameters_t * apex_inputs )`

Fill the APEX parameters structure with all the default parameters for APEX algorithms (pedometer, tilt)

Parameters

out	<i>apex_inputs</i>	Default input parameters. See
-----	--------------------	-------------------------------

See also

[inv\\_icm426xx\\_apex\\_parameters\\_t](#)

Returns

0 on success, negative value on error.

4.2.2.21 `int inv_icm426xx_init_tap_parameters_struct ( struct inv_icm426xx * s, inv_icm426xx_tap_parameters_t * tap_inputs )`

Fill the TAP parameters structure with all the default parameters for TAP algorithm.

Parameters

out	<i>tap_inputs</i>	Default input parameters. See
-----	-------------------	-------------------------------

See also

[inv\\_icm426xx\\_tap\\_parameters\\_t](#)

Returns

0 on success, negative value on error.

4.2.2.22 `int inv_icm426xx_set_apex_frequency ( struct inv_icm426xx * s, const ICM426XX_APEX_CONFIG0_DMP↵_ODR_t frequency )`

Configure DMP Output Data Rate for APEX algorithms (pedometer, tilt)



**Parameters**

in	<i>frequency</i>	The requested frequency.
----	------------------	--------------------------

**See also**[ICM426XX\\_APEX\\_CONFIG0\\_DMP\\_ODR\\_t](#)**Warning**

DMP\_ODR can change on the fly, and the DMP code will accommodate necessary modifications  
The user needs to take care to set Accel frequency  $\geq$  DMP frequency. This is a hard constraint since HW will not handle incorrect setting.

**Returns**

0 on success, negative value on error.

**4.2.2.23** `int inv_icm426xx_start_dmp ( struct inv_icm426xx * s )`

Start DMP for APEX algorithms.

**Returns**

0 on success, negative value on error.

### 4.3 Icm426xx driver extern functions

Extern functions for Icm426xx devices.

#### Files

- file [Icm426xxExtFunc.h](#)  
*Extern functions for Icm426xx devices.*

#### Functions

- void [inv\\_icm426xx\\_sleep\\_us](#) (uint32\_t us)  
*Hook for low-level high res system sleep() function to be implemented by upper layer ~100us resolution is sufficient.*
- uint64\_t [inv\\_icm426xx\\_get\\_time\\_us](#) (void)  
*Hook for low-level high res system get\_time() function to be implemented by upper layer Timer should be on 64bit with a 1 us resolution.*

#### 4.3.1 Detailed Description

Extern functions for Icm426xx devices.

#### 4.3.2 Function Documentation

##### 4.3.2.1 uint64\_t inv\_icm426xx\_get\_time\_us ( void )

Hook for low-level high res system get\_time() function to be implemented by upper layer Timer should be on 64bit with a 1 us resolution.

##### Parameters

out	<i>The</i>	current time in us
-----	------------	--------------------

##### 4.3.2.2 void inv\_icm426xx\_sleep\_us ( uint32\_t us )

Hook for low-level high res system sleep() function to be implemented by upper layer ~100us resolution is sufficient.

##### Parameters

in	<i>us</i>	number of us the calling thread should sleep
----	-----------	--

## 4.4 lcm426xx selftest

Low-level function to run selftest on a lcm426xx device.

### Files

- file [lcm426xxSelfTest.h](#)  
*Low-level function to run selftest on a lcm426xx device.*

### Functions

- int [inv\\_icm426xx\\_run\\_selftest](#) (struct [inv\\_icm426xx](#) \*s, int \*result)  
*Perform hardware self-test for Accel and Gyro.*
- int [inv\\_icm426xx\\_get\\_st\\_bias](#) (struct [inv\\_icm426xx](#) \*s, int st\_bias[6])  
*Retrieve bias collected by self-test.*
- int [inv\\_icm426xx\\_set\\_st\\_bias](#) (struct [inv\\_icm426xx](#) \*s, const int st\_bias[6])  
*Apply bias.*

#### 4.4.1 Detailed Description

Low-level function to run selftest on a lcm426xx device.

#### 4.4.2 Function Documentation

##### 4.4.2.1 int inv\_icm426xx\_get\_st\_bias ( struct inv\_icm426xx \* s, int st\_bias[6] )

Retrieve bias collected by self-test.

##### Parameters

out	<i>st_bias</i>	bias scaled by $2^{16}$ , accel is gee and gyro is dps. The buffer will be filled as below. Gyro LN mode X,Y,Z Accel LN mode X,Y,Z
-----	----------------	--

##### Returns

0 if success, error code if failure

##### 4.4.2.2 int inv\_icm426xx\_run\_selftest ( struct inv\_icm426xx \* s, int \* result )

Perform hardware self-test for Accel and Gyro.

##### Parameters

in	<i>result</i>	containing ACCEL_SUCCESS<<1   GYRO_SUCCESS so 3
----	---------------	---

**Returns**

0 if success, error code if failure

4.4.2.3 `int inv_icm426xx_set_st_bias ( struct inv_icm426xx * s, const int st_bias[6] )`

Apply bias.

**Parameters**

in	<i>st_bias</i>	bias scaled by $2^{16}$ , accel is gee and gyro is dps. The buffer must be filled as below. Gyro LN mode X,Y,Z Accel LN mode X,Y,Z
----	----------------	--

**Returns**

0 if success, error code if failure

## 4.5 lcm426xx driver transport

Low-level lcm426xx register access.

### Files

- file [lcm426xxTransport.h](#)  
*Low-level lcm426xx register access.*

### Classes

- struct [inv\\_icm426xx\\_serif](#)  
*basesensor serial interface*
- struct [inv\\_icm426xx\\_transport](#)  
*transport interface*

### Enumerations

### Functions

- int [inv\\_icm426xx\\_init\\_transport](#) (struct [inv\\_icm426xx](#) \*s)  
*Init cache variable.*
- int [inv\\_icm426xx\\_read\\_reg](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t reg, uint32\_t len, uint8\_t \*buf)  
*Reads data from a register on lcm426xx.*
- int [inv\\_icm426xx\\_write\\_reg](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t reg, uint32\_t len, const uint8\_t \*buf)  
*Writes data to a register on lcm426xx.*

#### 4.5.1 Detailed Description

Low-level lcm426xx register access.

#### 4.5.2 Function Documentation

##### 4.5.2.1 int [inv\\_icm426xx\\_init\\_transport](#) ( struct [inv\\_icm426xx](#) \* s )

Init cache variable.

##### Returns

0 in case of success, -1 for any error

##### 4.5.2.2 int [inv\\_icm426xx\\_read\\_reg](#) ( struct [inv\\_icm426xx](#) \* s, uint8\_t reg, uint32\_t len, uint8\_t \* buf )

Reads data from a register on lcm426xx.

**Parameters**

in	<i>reg</i>	register address to be read
in	<i>len</i>	number of byte to be read
out	<i>buf</i>	output data from the register

**Returns**

0 in case of success, -1 for any error

**4.5.2.3** `int inv_icm426xx_write_reg ( struct inv_icm426xx * s, uint8_t reg, uint32_t len, const uint8_t * buf )`

Writes data to a register on lcm426xx.

**Parameters**

in	<i>reg</i>	register address to be written
in	<i>len</i>	number of byte to be written
in	<i>buf</i>	input data to write

**Returns**

0 in case of success, -1 for any error

## Chapter 5

# Class Documentation

### 5.1 fifo\_header\_t Union Reference

Describe the content of the FIFO header.

```
#include <Icm426xxDefs.h>
```

#### 5.1.1 Detailed Description

Describe the content of the FIFO header.

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

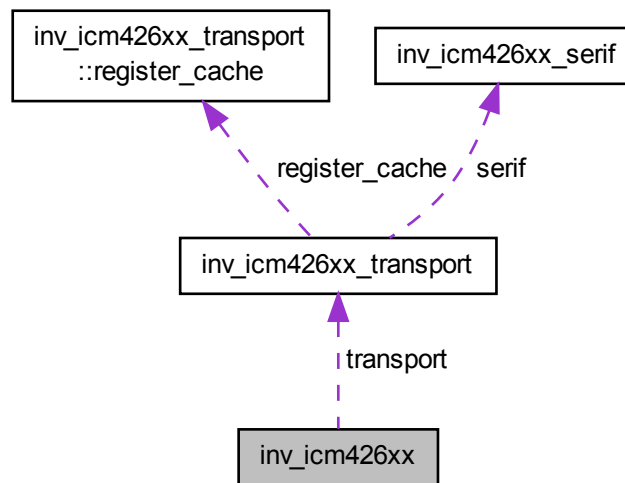
- [Icm426xxDefs.h](#)

### 5.2 inv\_icm426xx Struct Reference

Icm426xx driver states definition.

```
#include <Icm426xxDriver_HL.h>
```

Collaboration diagram for `inv_icm426xx`:



## Public Attributes

- struct [inv\\_icm426xx\\_transport](#) `transport`  
Warning : this field **MUST** be the first one of struct `icm426xx`.
- void(\* [sensor\\_event\\_cb](#))([inv\\_icm426xx\\_sensor\\_event\\_t](#) \*event)  
callback executed by `inv_icm426xx_get_data_from_fifo` function for each data packet extracted from fifo or `inv_icm426xx_get_data_from_registers` read data from register This field may be NULL if `inv_icm426xx_get_data_from_fifo/inv_icm426xx_get_data_from_registers` is not used by application.
- int [gyro\\_st\\_bias](#) [3]  
collected bias values (lsb) during self test
- int [st\\_result](#)  
Flag to keep track if self-test has been already run by storing acc and gyr results.
- uint8\_t [fifo\\_data](#) [[ICM426XX\\_FIFO\\_MIRRORING\\_SIZE](#)]  
FIFO mirroring memory area.
- uint8\_t [tmst\\_to\\_reg\\_en\\_cnt](#)  
internal counter to keep track of the timestamp to register access availability
- uint8\_t [dmp\\_is\\_on](#)  
DMP started status.
- uint64\_t [gyro\\_start\\_time\\_us](#)  
internal state needed to discard first gyro samples
- uint64\_t [accel\\_start\\_time\\_us](#)  
internal state needed to discard first accel samples
- uint8\_t [endianess\\_data](#)  
internal status of data endianess mode to report correctly data
- uint8\_t [fifo\\_highres\\_enabled](#)  
FIFO packets are 20 bytes long.
- [INV\\_ICM426XX\\_FIFO\\_CONFIG\\_t](#) `fifo_is_used`  
Data are get from FIFO or from sensor registers.



- `uint8_t wom_smd_mask`  
*This variable keeps track if wom or smd is enabled.*
- `uint8_t wom_enable`  
*This variable keeps track if wom is enabled.*
- `uint64_t gyro_power_off_tmst`  
*This variable keeps track of timestamp when gyro is power off.*
- `uint8_t acc_lp_avg`  
*Low-power averaging setting for accelerometer.*
- `uint8_t reserved`  
*reserved field*
- `uint8_t acc_ln_bw`  
*Low-noise filter bandwidth setting for accelerometer.*
- `uint8_t gyr_ln_bw`  
*Low-noise filter bandwidth setting for gyroscope.*

### 5.2.1 Detailed Description

Icm426xx driver states definition.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 INV\_ICM426XX\_FIFO\_CONFIG\_t inv\_icm426xx::fifo\_is\_used

Data are get from FIFO or from sensor registers.

By default Fifo is used

#### 5.2.2.2 void(\* inv\_icm426xx::sensor\_event\_cb)(inv\_icm426xx\_sensor\_event\_t \*event)

callback executed by `inv_icm426xx_get_data_from_fifo` function for each data packet extracted from fifo or `inv_icm426xx_get_data_from_registers` read data from register This field may be NULL if `inv_icm426xx_get_data_from_fifo/inv_icm426xx_get_data_from_registers` is not used by application.

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

- [Icm426xxDriver\\_HL.h](#)

## 5.3 inv\_icm426xx\_apex\_parameters Struct Reference

Icm426xx APEX inputs parameters definition.

```
#include <Icm426xxDriver_HL_apex.h>
```

## Public Attributes

- ICM426XX\_APEX\_CONFIG2\_PEDO\_AMP\_TH\_t [pedo\\_amp\\_th](#)  
*Peak threshold value to be considered as a valid step (mg)*
- uint8\_t [pedo\\_step\\_cnt\\_th](#)  
*Minimum number of steps that must be detected before the pedometer step count begins incrementing.*
- uint8\_t [pedo\\_step\\_det\\_th](#)  
*Minimum number of low latency steps that must be detected before the pedometer step count begins incrementing.*
- ICM426XX\_APEX\_CONFIG3\_PEDO\_SB\_TIMER\_TH\_t [pedo\\_sb\\_timer\\_th](#)  
*Duration of non-walk to exit the current walk mode, pedo\_step\_cnt\_th number of steps must again be detected before step count starts to increase.*
- ICM426XX\_APEX\_CONFIG3\_PEDO\_HI\_ENRGY\_TH\_t [pedo\\_hi\\_enrgy\\_th](#)  
*Threshold to improve run detection if not steps are counted while running.*
- ICM426XX\_APEX\_CONFIG4\_TILT\_WAIT\_TIME\_t [tilt\\_wait\\_time](#)  
*Number of accelerometer samples to wait before triggering tilt event.*
- ICM426XX\_APEX\_CONFIG1\_DMP\_POWER\_SAVE\_TIME\_t [power\\_save\\_time](#)  
*The time after which DMP goes in power save mode according to the DMP ODR configured.*
- ICM426XX\_APEX\_CONFIG0\_DMP\_POWER\_SAVE\_t [power\\_save](#)  
*Power save mode for APEX algorithms.*
- ICM426XX\_APEX\_CONFIG9\_SENSITIVITY\_MODE\_t [sensitivity\\_mode](#)  
*Sensitivity mode Normal(0) or Slow walk(1).*
- ICM426XX\_APEX\_CONFIG4\_R2W\_SLEEP\_TIME\_OUT\_t [r2w\\_sleep\\_time\\_out](#)  
*Time out for a sleep detection.*
- ICM426XX\_APEX\_CONFIG5\_R2W\_MOUNTING\_MATRIX\_t [r2w\\_mounting\\_matrix](#)  
*Mounting matrix, chip to device frame.*
- ICM426XX\_APEX\_CONFIG6\_R2W\_SLEEP\_GEST\_DELAY\_t [r2w\\_gest\\_delay](#)  
*Detection window for a sleep gesture detection.*
- ICM426XX\_APEX\_CONFIG1\_LOW\_ENERGY\_AMP\_TH\_t [low\\_energy\\_amp\\_th](#)  
*Peak threshold value to be considered as a valid step (mg) in Slow walk mode.*

### 5.3.1 Detailed Description

lcm426xx APEX inputs parameters definition.

### 5.3.2 Member Data Documentation

#### 5.3.2.1 ICM426XX\_APEX\_CONFIG0\_DMP\_POWER\_SAVE\_t inv\_lcm426xx\_apex\_parameters::power\_save

Power save mode for APEX algorithms.

This mode will put APEX features into sleep mode, leaving only the WOM running to wake-up the DMP

#### 5.3.2.2 ICM426XX\_APEX\_CONFIG6\_R2W\_SLEEP\_GEST\_DELAY\_t inv\_lcm426xx\_apex\_parameters::r2w\_gest\_delay

Detection window for a sleep gesture detection.

## 5.3.2.3 ICM426XX\_APEX\_CONFIG5\_R2W\_MOUNTING\_MATRIX\_t inv\_icm426xx\_apex\_parameters::r2w\_mounting\_matrix

Mounting matrix, chip to device frame.

## 5.3.2.4 ICM426XX\_APEX\_CONFIG4\_R2W\_SLEEP\_TIME\_OUT\_t inv\_icm426xx\_apex\_parameters::r2w\_sleep\_time\_out

Time out for a sleep detection.

## 5.3.2.5 ICM426XX\_APEX\_CONFIG9\_SENSITIVITY\_MODE\_t inv\_icm426xx\_apex\_parameters::sensitivity\_mode

Sensitivity mode Normal(0) or Slow walk(1).

The Slow walk mode improve the slow walk detection (<1Hz) but in return the number of false detection might be increase.

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

- [Icm426xxDriver\\_HL\\_apex.h](#)

## 5.4 inv\_icm426xx\_apex\_step\_activity Struct Reference

APEX pedometer outputs.

```
#include <Icm426xxDriver_HL_apex.h>
```

### Public Attributes

- uint16\_t [step\\_cnt](#)  
*Number of steps taken.*
- uint8\_t [step\\_cadence](#)  
*Walk/run cadency in number of samples.*
- uint8\_t [activity\\_class](#)  
*Detected activity unknown (0), walk (1) or run (2)*

### 5.4.1 Detailed Description

APEX pedometer outputs.

### 5.4.2 Member Data Documentation

## 5.4.2.1 uint8\_t inv\_icm426xx\_apex\_step\_activity::step\_cadence

Walk/run cadency in number of samples.

Format is u6.2. E.g, At 50Hz and 2Hz walk frequency, if the cadency is 25 samples. The register will output 100.

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

- [Icm426xxDriver\\_HL\\_apex.h](#)

## 5.5 `inv_icm426xx_interrupt_parameter_t` Struct Reference

Icm426xx set of interrupt enable flag.

```
#include <Icm426xxDriver_HL.h>
```

### 5.5.1 Detailed Description

Icm426xx set of interrupt enable flag.

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

- [Icm426xxDriver\\_HL.h](#)

## 5.6 `inv_icm426xx_sensor_event_t` Struct Reference

Sensor event structure definition.

```
#include <Icm426xxDriver_HL.h>
```

### 5.6.1 Detailed Description

Sensor event structure definition.

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

- [Icm426xxDriver\\_HL.h](#)

## 5.7 `inv_icm426xx_serif` Struct Reference

basesensor serial interface

```
#include <Icm426xxTransport.h>
```

### 5.7.1 Detailed Description

basesensor serial interface

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

- [Icm426xxTransport.h](#)

## 5.8 inv\_icm426xx\_tap\_data Struct Reference

TAP outputs.

```
#include <Icm426xxDriver_HL_apex.h>
```

### Public Attributes

- ICM426XX\_APEX\_DATA4\_TAP\_NUM\_t [tap\\_num](#)  
*Detects single (1) or double (2) tap.*
- ICM426XX\_APEX\_DATA4\_TAP\_AXIS\_t [tap\\_axis](#)  
*Axis along which tap has been detected.*
- ICM426XX\_APEX\_DATA4\_TAP\_DIR\_t [tap\\_dir](#)  
*Direction of the tap, either +axis (1) or -axis (0)*
- uint8\_t [double\\_tap\\_timing](#)  
*Timing between both taps of a double tap expressed in 1/16th of odr in ms (e.g At 500Hz, 2 means 64ms between each tap)*

### 5.8.1 Detailed Description

TAP outputs.

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

- [lcm426xxDriver\\_HL\\_apex.h](#)

## 5.9 inv\_icm426xx\_tap\_parameters\_t Struct Reference

Icm426xx TAP inputs parameters definition.

```
#include <Icm426xxDriver_HL_apex.h>
```

### Public Attributes

- uint8\_t [min\\_jerk\\_thr](#)  
*Minimum Jerk Threshold.*
- ICM426XX\_APEX\_CONFIG7\_TAP\_MAX\_PEAK\_TOL\_t [max\\_peak\\_tol](#)  
*Maximum peak tolerance.*
- ICM426XX\_APEX\_CONFIG8\_TAP\_TMAX\_t [tmax](#)  
*Tap measurement window.*
- ICM426XX\_APEX\_CONFIG8\_TAP\_TAVG\_t [tavg](#)  
*Energy measumerement window.*
- ICM426XX\_APEX\_CONFIG8\_TAP\_TMIN\_t [tmin](#)  
*Single tap window.*

### 5.9.1 Detailed Description

Icm426xx TAP inputs parameters definition.

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

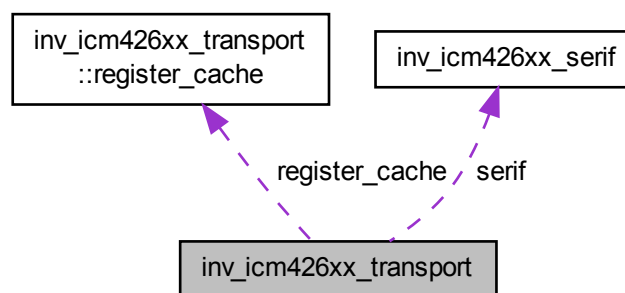
- [Icm426xxDriver\\_HL\\_apex.h](#)

## 5.10 inv\_icm426xx\_transport Struct Reference

transport interface

```
#include <Icm426xxTransport.h>
```

Collaboration diagram for inv\_icm426xx\_transport:



### Classes

- struct [register\\_cache](#)  
*Contains mirrored values of some IP registers.*

### Public Attributes

- struct [inv\\_icm426xx\\_serif](#) `serif`  
*Warning : this field MUST be the first one of struct [inv\\_icm426xx\\_transport](#).*
- struct [inv\\_icm426xx\\_transport::register\\_cache](#) `register_cache`  
*Store mostly used register values on SRAM.*

### 5.10.1 Detailed Description

transport interface

## 5.10.2 Member Data Documentation

### 5.10.2.1 struct inv\_icm426xx\_transport::register\_cache inv\_icm426xx\_transport::register\_cache

Store mostly used register values on SRAM.

MPUREG\_OTP\_SEC\_STATUS\_B1 and MPUREG\_INT\_STATUS registers are read before the cache has a chance to be initialized. Therefore, these registers shall never be added to the cache Registers from bank 1,2,3 or 4 shall never be added to the cache

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

- [lcm426xxTransport.h](#)

## 5.11 recover\_regs Struct Reference

Contains the current register values.

### 5.11.1 Detailed Description

Contains the current register values.

Used to reapply values after the ST procedure

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

- [lcm426xxSelfTest.c](#)

## 5.12 inv\_icm426xx\_transport::register\_cache Struct Reference

Contains mirrored values of some IP registers.

```
#include <Icm426xxTransport.h>
```

### Public Attributes

- [uint8\\_t intf\\_cfg\\_1\\_reg](#)  
*INTF\_CONFIG1, Bank: 0, Address: 0x4D.*
- [uint8\\_t pwr\\_mngt\\_0\\_reg](#)  
*PWR\_MGMT\_0, Bank: 0, Address: 0x4E.*
- [uint8\\_t gyro\\_cfg\\_0\\_reg](#)  
*GYRO\_CONFIG0, Bank: 0, Address: 0x4F.*
- [uint8\\_t accel\\_cfg\\_0\\_reg](#)  
*ACCEL\_CONFIG0, Bank: 0, Address: 0x50.*
- [uint8\\_t tmst\\_cfg\\_reg](#)  
*TMST\_CONFIG, Bank: 0, Address: 0x54.*
- [uint8\\_t bank\\_sel\\_reg](#)  
*MPUREG\_REG\_BANK\_SEL, All banks, Address 0x76.*

### 5.12.1 Detailed Description

Contains mirrored values of some IP registers.

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

- [lcm426xxTransport.h](#)





## Chapter 6

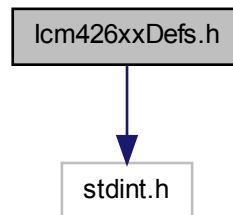
# File Documentation

### 6.1 lcm426xxDefs.h File Reference

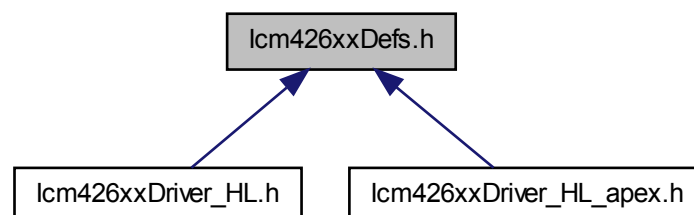
File exposing the device register map.

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

Include dependency graph for lcm426xxDefs.h:



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



## Classes

- union [fifo\\_header\\_t](#)

*Describe the content of the FIFO header.*

## Macros

- #define [BIT\\_APEX\\_DATA4\\_TAP\\_NUM\\_POS](#) 3

*TAP status flags: non-zero value - tap detected bit0 - positive or negative edge bit1 and 2 - axis detected : 0-X ; 1-Y ; 2-Z bit3 and 4 - tap type : 1-single ; 2 -double.*

## Enumerations

### 6.1.1 Detailed Description

File exposing the device register map.

### 6.1.2 Enumeration Type Documentation

#### 6.1.2.1 enum ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_t

Accelerometer FSR selection.

##### Enumerator

**ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_2g** 2g  
**ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_4g** 4g  
**ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_8g** 8g  
**ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_16g** 16g

#### 6.1.2.2 enum ICM426XX\_ACCEL\_CONFIG0\_ODR\_t

Accelerometer ODR selection.

##### Enumerator

**ICM426XX\_ACCEL\_CONFIG0\_ODR\_500\_HZ** 500 Hz (2 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_5625\_HZ** 1.5625 Hz (640 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_3\_125\_HZ** 3.125 Hz (320 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_6\_25\_HZ** 6.25 Hz (160 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_12\_5\_HZ** 12.5 Hz (80 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_25\_HZ** 25 Hz (40 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_50\_HZ** 50 Hz (20 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_100\_HZ** 100 Hz (10 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_200\_HZ** 200 Hz (5 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_KHZ** 1 KHz (1 ms)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_2\_KHZ** 2 KHz (500 us)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_4\_KHZ** 4 KHz (250 us)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_8\_KHZ** 8 KHz (125 us)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_16\_KHZ** 16 KHz (62.5 us)  
**ICM426XX\_ACCEL\_CONFIG0\_ODR\_32\_KHZ** 32 KHz (31.25 us)

## 6.1.2.3 enum ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_t

DMP ODR selection.

Enumerator

**ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_25Hz** 25Hz (40ms)  
**ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_50Hz** 50Hz (20ms)  
**ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_RESERVED** Reserved.

## 6.1.2.4 enum ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_t

Gyroscope FSR selection.

Enumerator

**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_16dps** 16dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_31dps** 31dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_62dps** 62dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_125dps** 125dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_250dps** 250dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_500dps** 500dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_1000dps** 1000dps  
**ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_2000dps** 2000dps

## 6.1.2.5 enum ICM426XX\_GYRO\_CONFIG0\_ODR\_t

Gyroscope ODR selection.

Enumerator

**ICM426XX\_GYRO\_CONFIG0\_ODR\_500\_HZ** 500 Hz (2 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_12\_5\_HZ** 12.5 Hz (80 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_25\_HZ** 25 Hz (40 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_50\_HZ** 50 Hz (20 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_100\_HZ** 100 Hz (10 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_200\_HZ** 200 Hz (5 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_1\_KHZ** 1 KHz (1 ms)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_2\_KHZ** 2 KHz (500 us)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_4\_KHZ** 4 KHz (250 us)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_8\_KHZ** 8 KHz (125 us)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_16\_KHZ** 16 KHz (62.5 us)  
**ICM426XX\_GYRO\_CONFIG0\_ODR\_32\_KHZ** 32 KHz (31.25 us)

## 6.1.2.6 enum ICM426XX\_OIS1\_CONFIG1\_DEC\_t

OIS1 rate selection (base clock fixed by OTP divided by decimator value)

Enumerator

```

ICM426XX_OIS1_CONFIG1_DEC_1 OTP_OIS_clock / 1.
ICM426XX_OIS1_CONFIG1_DEC_2 OTP_OIS_clock / 2.
ICM426XX_OIS1_CONFIG1_DEC_4 OTP_OIS_clock / 4.
ICM426XX_OIS1_CONFIG1_DEC_8 OTP_OIS_clock / 8.
ICM426XX_OIS1_CONFIG1_DEC_16 OTP_OIS_clock / 16.
ICM426XX_OIS1_CONFIG1_DEC_32 OTP_OIS_clock / 32.

```

## 6.1.2.7 enum ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_t

OIS1 accelerometer FSR selection.

Enumerator

```

ICM426XX_OIS1_CONFIG2_ACCEL_FS_SEL_2g 2g
ICM426XX_OIS1_CONFIG2_ACCEL_FS_SEL_4g 4g
ICM426XX_OIS1_CONFIG2_ACCEL_FS_SEL_8g 8g
ICM426XX_OIS1_CONFIG2_ACCEL_FS_SEL_16g 16g

```

## 6.1.2.8 enum ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_t

OIS1 gyroscope FSR selection.

Enumerator

```

ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_16dps 15.625 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_31dps 31.25 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_62dps 62.5 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_125dps 125 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_250dps 250 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_500dps 500 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_1000dps 1000 dps
ICM426XX_OIS1_CONFIG2_GYRO_FS_SEL_2000dps 2000 dps

```

## 6.1.2.9 enum ICM426XX\_OIS2\_CONFIG1\_DEC\_t

OIS2 rate selection (base clock fixed by OTP divided by decimator value)

Enumerator

```

ICM426XX_OIS2_CONFIG1_DEC_1 OTP_OIS_clock / 1.
ICM426XX_OIS2_CONFIG1_DEC_2 OTP_OIS_clock / 2.
ICM426XX_OIS2_CONFIG1_DEC_4 OTP_OIS_clock / 4.
ICM426XX_OIS2_CONFIG1_DEC_8 OTP_OIS_clock / 8.
ICM426XX_OIS2_CONFIG1_DEC_16 OTP_OIS_clock / 16.
ICM426XX_OIS2_CONFIG1_DEC_32 OTP_OIS_clock / 32.

```

## 6.1.2.10 enum ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_t

OIS2 accelerometer FSR selection.

Enumerator

```

ICM426XX_OIS2_CONFIG2_ACCEL_FS_SEL_2g  2g
ICM426XX_OIS2_CONFIG2_ACCEL_FS_SEL_4g  4g
ICM426XX_OIS2_CONFIG2_ACCEL_FS_SEL_8g  8g
ICM426XX_OIS2_CONFIG2_ACCEL_FS_SEL_16g 16g

```

## 6.1.2.11 enum ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_t

OIS2 gyroscope FSR selection.

Enumerator

```

ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_16dps 15.625 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_31dps 31.25 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_62dps 62.5 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_125dps 125 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_250dps 250 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_500dps 500 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_1000dps 1000 dps
ICM426XX_OIS2_CONFIG2_GYRO_FS_SEL_2000dps 2000 dps

```

## 6.2 Icm426xxDriver\_HL.h File Reference

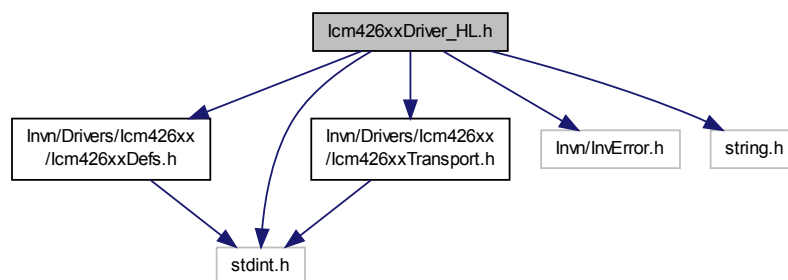
High-level function to setup an Icm426xx device.

```

#include "Invn/Drivers/Icm426xx/Icm426xxDefs.h"
#include "Invn/Drivers/Icm426xx/Icm426xxTransport.h"
#include "Invn/InvError.h"
#include <stdint.h>
#include <string.h>

```

Include dependency graph for Icm426xxDriver\_HL.h:



## Classes

- struct [inv\\_icm426xx\\_sensor\\_event\\_t](#)  
*Sensor event structure definition.*
- struct [inv\\_icm426xx](#)  
*Icm426xx driver states definition.*
- struct [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#)  
*Icm426xx set of interrupt enable flag.*

## Macros

- #define [INV\\_ICM426XX\\_LIGHTWEIGHT\\_DRIVER](#) 0  
*Lighthen driver logic by stripping out procedures on transitions.*
- #define [PLL\\_SCALE\\_FACTOR\\_Q24](#) (1UL<<24)  
*Scale factor and max ODR Dependant of chip.*
- #define [ACCEL\\_CONFIG0\\_FS\\_SEL\\_MAX](#) ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_16g  
*Max FSR values for accel and gyro Dependant of chip.*
- #define [RTC\\_SUPPORTED](#) 0  
*RTC Support flag Define whether the RTC mode is supported Dependant of chip.*
- #define [ICM426XX\\_FIFO\\_MIRRORING\\_SIZE](#) 16 \* 129  
*Icm426xx maximum buffer size mirrored from FIFO at polling time.*
- #define [ICM426XX\\_DEFAULT\\_WOM\\_THS\\_MG](#) 52>>2 /\* = 52mg/4 \*/  
*Default value for the WOM threshold Resolution of the threshold is ~ = 4mg.*
- #define [ICM426XX\\_ACC\\_STARTUP\\_TIME\\_US](#) 20000U  
*Icm426xx Accelerometer start-up time before having correct data.*
- #define [ICM426XX\\_GYR\\_STARTUP\\_TIME\\_US](#) 60000U  
*Icm426xx Gyroscope start-up time before having correct data.*

## Enumerations

## Functions

- int [inv\\_icm426xx\\_set\\_reg\\_bank](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t bank)  
*Set register bank index.*
- int [inv\\_icm426xx\\_init](#) (struct [inv\\_icm426xx](#) \*s, struct [inv\\_icm426xx\\_serif](#) \*serif, void(\*sensor\_event\_cb)([inv\\_icm426xx\\_sensor\\_event\\_t](#) \*event))  
*Configure the serial interface used to access the device and execute hardware initialization.*
- int [inv\\_icm426xx\\_device\\_reset](#) (struct [inv\\_icm426xx](#) \*s)  
*Perform a soft reset of the device.*
- int [inv\\_icm426xx\\_get\\_who\\_am\\_i](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t \*who\_am\_i)  
*return WHOAMI value*
- int [inv\\_icm426xx\\_force\\_clock\\_source](#) (struct [inv\\_icm426xx](#) \*s, ICM426XX\_INTF\_CONFIG1\_ACCEL\_LP\_↵ CLK\_t clk\_src)  
*Configure Accel clock source.*
- int [inv\\_icm426xx\\_enable\\_accel\\_low\\_power\\_mode](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable/put accel in low power mode.*
- int [inv\\_icm426xx\\_enable\\_accel\\_low\\_noise\\_mode](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable/put accel in low noise mode.*
- int [inv\\_icm426xx\\_disable\\_accel](#) (struct [inv\\_icm426xx](#) \*s)

- Disable all 3 axes of accel.*

  - int [inv\\_icm426xx\\_enable\\_gyro\\_low\\_noise\\_mode](#) (struct [inv\\_icm426xx](#) \*s)

*Enable/put gyro in low noise mode.*
- int [inv\\_icm426xx\\_disable\\_gyro](#) (struct [inv\\_icm426xx](#) \*s)

*Disable all 3 axes of gyro.*
- int [inv\\_icm426xx\\_enable\\_fsyc](#) (struct [inv\\_icm426xx](#) \*s)

*Enable fsync tagging fonctionnality.*
- int [inv\\_icm426xx\\_disable\\_fsyc](#) (struct [inv\\_icm426xx](#) \*s)

*Disable fsync tagging fonctionnality.*
- int [inv\\_icm426xx\\_configure\\_timestamp\\_resolution](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_TMST\\_CONFIG\\_RESOL\\_t](#) resol)

*Configure timestamp resolution from FIFO.*
- int [inv\\_icm426xx\\_set\\_config\\_ibi](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Configure which interrupt source can trigger ibi interruptions.*
- int [inv\\_icm426xx\\_get\\_config\\_ibi](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Retrieve interrupts configuration.*
- int [inv\\_icm426xx\\_set\\_config\\_int1](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Configure which interrupt source can trigger INT1.*
- int [inv\\_icm426xx\\_get\\_config\\_int1](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Retrieve interrupts configuration.*
- int [inv\\_icm426xx\\_set\\_config\\_int2](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Configure which interrupt source can trigger INT2.*
- int [inv\\_icm426xx\\_get\\_config\\_int2](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_interrupt\\_parameter\\_t](#) \*interrupt\_↵ to\_configure)

*Retrieve interrupts configuration.*
- int [inv\\_icm426xx\\_get\\_data\\_from\\_registers](#) (struct [inv\\_icm426xx](#) \*s)

*Read all registers containing data (temperature, accelerometer and gyroscope).*
- int [inv\\_icm426xx\\_get\\_data\\_from\\_fifo](#) (struct [inv\\_icm426xx](#) \*s)

*Read all available packets from the FIFO.*
- uint32\_t [inv\\_icm426xx\\_convert\\_odr\\_bitfield\\_to\\_us](#) (uint32\_t odr\_bitfield)

*Converts ICM426XX\_ACCEL\_CONFIG0\_ODR\_t or ICM426XX\_GYRO\_CONFIG0\_ODR\_t enums to period expressed in us.*
- int [inv\\_icm426xx\\_set\\_accel\\_frequency](#) (struct [inv\\_icm426xx](#) \*s, const [ICM426XX\\_ACCEL\\_CONFIG0\\_ODR\\_t](#) frequency)

*Configure accel Output Data Rate.*
- int [inv\\_icm426xx\\_set\\_gyro\\_frequency](#) (struct [inv\\_icm426xx](#) \*s, const [ICM426XX\\_GYRO\\_CONFIG0\\_ODR\\_t](#) frequency)

*Configure gyro Output Data Rate.*
- int [inv\\_icm426xx\\_set\\_accel\\_fsr](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_ACCEL\\_CONFIG0\\_FS\\_SEL\\_t](#) accel\_↵ \_fsr\_g)

*Set accel full scale range.*
- int [inv\\_icm426xx\\_get\\_accel\\_fsr](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_ACCEL\\_CONFIG0\\_FS\\_SEL\\_t](#) \*accel\_↵ \_fsr\_g)

*Access accel full scale range.*
- int [inv\\_icm426xx\\_set\\_gyro\\_fsr](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_GYRO\\_CONFIG0\\_FS\\_SEL\\_t](#) gyro\_fsr\_↵ \_dps)

*Set gyro full scale range.*

- int [inv\\_icm426xx\\_get\\_gyro\\_fsr](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_GYRO\\_CONFIG0\\_FS\\_SEL\\_t](#) \*gyro\_fsr\_dps)  
*Access gyro full scale range.*
- int [inv\\_icm426xx\\_set\\_accel\\_lp\\_avg](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_GYRO\\_ACCEL\\_CONFIG0\\_ACCEL\\_EL\\_FILT\\_AVG\\_t](#) acc\_avg)  
*Set accel Low-Power averaging value.*
- int [inv\\_icm426xx\\_set\\_accel\\_ln\\_bw](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_GYRO\\_ACCEL\\_CONFIG0\\_ACCEL\\_EL\\_FILT\\_BW\\_t](#) acc\_bw)  
*Set accel Low-Noise bandwidth value.*
- int [inv\\_icm426xx\\_set\\_gyro\\_ln\\_bw](#) (struct [inv\\_icm426xx](#) \*s, [ICM426XX\\_GYRO\\_ACCEL\\_CONFIG0\\_GYRO\\_FILT\\_BW\\_t](#) gyr\_bw)  
*Set gyro Low-Noise bandwidth value.*
- int [inv\\_icm426xx\\_reset\\_fifo](#) (struct [inv\\_icm426xx](#) \*s)  
*reset ICM426XX fifo*
- int [inv\\_icm426xx\\_enable\\_timestamp\\_to\\_register](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable the 20bits-timestamp register access to read in a reliable way the strobed timestamp.*
- int [inv\\_icm426xx\\_disable\\_timestamp\\_to\\_register](#) (struct [inv\\_icm426xx](#) \*s)  
*Disable the 20bits-timestamp register access.*
- int [inv\\_icm426xx\\_get\\_current\\_timestamp](#) (struct [inv\\_icm426xx](#) \*s, [uint32\\_t](#) \*icm\_time)  
*Get the timestamp value of icm426xx from register.*
- int [inv\\_icm426xx\\_enable\\_clkin\\_rtc](#) (struct [inv\\_icm426xx](#) \*s, [uint8\\_t](#) enable)  
*Enable or disable CLKIN/RTC capability.*
- int [inv\\_icm426xx\\_get\\_clkin\\_rtc\\_status](#) (struct [inv\\_icm426xx](#) \*s)  
*Get CLKIN/RTC feature status.*
- int [inv\\_icm426xx\\_enable\\_high\\_resolution\\_fifo](#) (struct [inv\\_icm426xx](#) \*s)  
*Enable 20 bits raw acc and raw gyr data in fifo.*
- int [inv\\_icm426xx\\_disable\\_high\\_resolution\\_fifo](#) (struct [inv\\_icm426xx](#) \*s)  
*Disable 20 bits raw acc and raw gyr data in fifo.*
- int [inv\\_icm426xx\\_configure\\_fifo](#) (struct [inv\\_icm426xx](#) \*s, [INV\\_ICM426XX\\_FIFO\\_CONFIG\\_t](#) fifo\_config)  
*Configure Fifo to select the way data are gathered.*
- int [inv\\_icm426xx\\_configure\\_fifo\\_wm](#) (struct [inv\\_icm426xx](#) \*s, [uint16\\_t](#) wm)  
*Configure Fifo watermark (also refered to as fifo threshold)*
- [uint32\\_t](#) [inv\\_icm426xx\\_get\\_fifo\\_timestamp\\_resolution\\_us\\_q24](#) (struct [inv\\_icm426xx](#) \*s)  
*Get FIFO timestamp resolution.*
- [uint32\\_t](#) [inv\\_icm426xx\\_get\\_reg\\_timestamp\\_resolution\\_us\\_q24](#) (struct [inv\\_icm426xx](#) \*s)  
*Get register timestamp resolution.*
- const char \* [inv\\_icm426xx\\_get\\_version](#) (void)  
*Return driver version x.y.z-suffix as a char array.*

## 6.2.1 Detailed Description

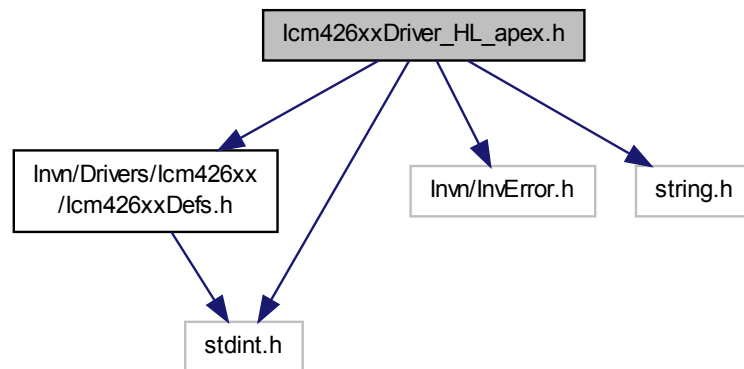
High-level function to setup an Icm426xx device.



## 6.3 lcm426xxDriver\_HL\_apex.h File Reference

High-level function to setup an lcm426xx device.

```
#include "Invn/Drivers/Icm426xx/Icm426xxDefs.h"
#include "Invn/InvError.h"
#include <stdint.h>
#include <string.h>
Include dependency graph for lcm426xxDriver_HL_apex.h:
```



### Classes

- struct [inv\\_icm426xx\\_tap\\_parameters\\_t](#)  
*lcm426xx TAP inputs parameters definition.*
- struct [inv\\_icm426xx\\_apex\\_parameters](#)  
*lcm426xx APEX inputs parameters definition.*
- struct [inv\\_icm426xx\\_apex\\_step\\_activity](#)  
*APEX pedometer outputs.*
- struct [inv\\_icm426xx\\_tap\\_data](#)  
*TAP outputs.*

### Typedefs

- typedef struct [inv\\_icm426xx\\_apex\\_parameters](#) [inv\\_icm426xx\\_apex\\_parameters\\_t](#)  
*lcm426xx APEX inputs parameters definition.*
- typedef struct [inv\\_icm426xx\\_apex\\_step\\_activity](#) [inv\\_icm426xx\\_apex\\_step\\_activity\\_t](#)  
*APEX pedometer outputs.*
- typedef struct [inv\\_icm426xx\\_tap\\_data](#) [inv\\_icm426xx\\_tap\\_data\\_t](#)  
*TAP outputs.*

## Functions

- `int inv_icm426xx_configure_smd_wom` (struct `inv_icm426xx` \*s, const uint8\_t x\_th, const uint8\_t y\_th, const uint8\_t z\_th, ICM426XX\_SMD\_CONFIG\_WOM\_INT\_MODE\_t wom\_int, ICM426XX\_SMD\_CONFIG\_WOM\_MODE\_t wom\_mode)  
*Configure Wake On Motion and SMD thresholds.*
- `int inv_icm426xx_enable_wom` (struct `inv_icm426xx` \*s)  
*Enable Wake On Motion.*
- `int inv_icm426xx_disable_wom` (struct `inv_icm426xx` \*s)  
*Disable Wake On Motion.*
- `int inv_icm426xx_enable_smd` (struct `inv_icm426xx` \*s)  
*Enable Significant Motion Detection.*
- `int inv_icm426xx_disable_smd` (struct `inv_icm426xx` \*s)  
*Disable Significant Motion Detection.*
- `int inv_icm426xx_init_tap_parameters_struct` (struct `inv_icm426xx` \*s, `inv_icm426xx_tap_parameters_t` \*tap\_inputs)  
*Fill the TAP parameters structure with all the default parameters for TAP algorithm.*
- `int inv_icm426xx_configure_tap_parameters` (struct `inv_icm426xx` \*s, const `inv_icm426xx_tap_parameters_t` \*tap\_inputs)  
*Configure TAP.*
- `int inv_icm426xx_get_tap_parameters` (struct `inv_icm426xx` \*s, `inv_icm426xx_tap_parameters_t` \*tap\_params)  
*Returns current TAP parameters.*
- `int inv_icm426xx_enable_tap` (struct `inv_icm426xx` \*s)  
*Enable TAP.*
- `int inv_icm426xx_disable_tap` (struct `inv_icm426xx` \*s)  
*Disable TAP.*
- `int inv_icm426xx_init_apex_parameters_struct` (struct `inv_icm426xx` \*s, `inv_icm426xx_apex_parameters_t` \*apex\_inputs)  
*Fill the APEX parameters structure with all the default parameters for APEX algorithms (pedometer, tilt)*
- `int inv_icm426xx_configure_apex_parameters` (struct `inv_icm426xx` \*s, const `inv_icm426xx_apex_parameters_t` \*apex\_inputs)  
*Configures DMP parameters for APEX algorithms (pedometer, tilt).*
- `int inv_icm426xx_get_apex_parameters` (struct `inv_icm426xx` \*s, `inv_icm426xx_apex_parameters_t` \*apex\_params)  
*Returns current DMP parameters for APEX algorithms (pedometer, tilt).*
- `int inv_icm426xx_set_apex_frequency` (struct `inv_icm426xx` \*s, const ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_t frequency)  
*Configure DMP Output Data Rate for APEX algorithms (pedometer, tilt)*
- `int inv_icm426xx_start_dmp` (struct `inv_icm426xx` \*s)  
*Start DMP for APEX algorithms.*
- `int inv_icm426xx_enable_apex_pedometer` (struct `inv_icm426xx` \*s)  
*Enable APEX algorithm Pedometer.*
- `int inv_icm426xx_disable_apex_pedometer` (struct `inv_icm426xx` \*s)  
*Disable APEX algorithm Pedometer.*
- `int inv_icm426xx_enable_apex_r2w` (struct `inv_icm426xx` \*s)  
*Enable APEX algorithm Raise to wake.*
- `int inv_icm426xx_disable_apex_r2w` (struct `inv_icm426xx` \*s)  
*Disable APEX algorithm Raise to wake.*
- `int inv_icm426xx_enable_apex_tilt` (struct `inv_icm426xx` \*s)  
*Enable APEX algorithm Tilt.*
- `int inv_icm426xx_disable_apex_tilt` (struct `inv_icm426xx` \*s)

*Disable APEX algorithm Tilt.*

- int [inv\\_icm426xx\\_get\\_apex\\_data\\_activity](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_apex\\_step\\_activity\\_t](#) \*apex\_activity)

*Retrieve APEX pedometer outputs and format them.*

- int [inv\\_icm426xx\\_get\\_tap\\_data](#) (struct [inv\\_icm426xx](#) \*s, [inv\\_icm426xx\\_tap\\_data\\_t](#) \*tap\_data)

*Retrieve tap outputs.*

### 6.3.1 Detailed Description

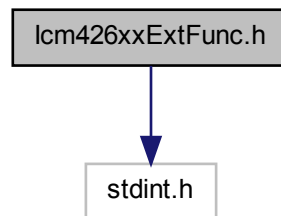
High-level function to setup an lcm426xx device.

## 6.4 lcm426xxExtFunc.h File Reference

Extern functions for lcm426xx devices.

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

Include dependency graph for lcm426xxExtFunc.h:



### Functions

- void [inv\\_icm426xx\\_sleep\\_us](#) (uint32\_t us)

*Hook for low-level high res system sleep() function to be implemented by upper layer ~100us resolution is sufficient.*

- uint64\_t [inv\\_icm426xx\\_get\\_time\\_us](#) (void)

*Hook for low-level high res system get\_time() function to be implemented by upper layer Timer should be on 64bit with a 1 us resolution.*

### 6.4.1 Detailed Description

Extern functions for lcm426xx devices.

## 6.5 lcm426xxSelfTest.h File Reference

Low-level function to run selftest on a lcm426xx device.

```
#include "Invn/InvExport.h"
```

Include dependency graph for lcm426xxSelfTest.h:



### Functions

- int [inv\\_icm426xx\\_run\\_selftest](#) (struct [inv\\_icm426xx](#) \*s, int \*result)  
*Perform hardware self-test for Accel and Gyro.*
- int [inv\\_icm426xx\\_get\\_st\\_bias](#) (struct [inv\\_icm426xx](#) \*s, int st\_bias[6])  
*Retrieve bias collected by self-test.*
- int [inv\\_icm426xx\\_set\\_st\\_bias](#) (struct [inv\\_icm426xx](#) \*s, const int st\_bias[6])  
*Apply bias.*

### 6.5.1 Detailed Description

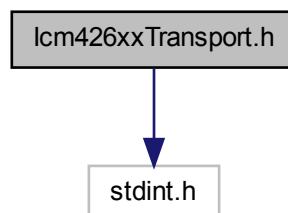
Low-level function to run selftest on a lcm426xx device.

## 6.6 lcm426xxTransport.h File Reference

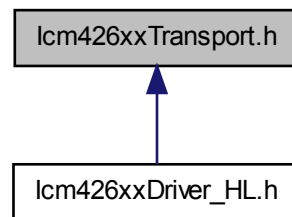
Low-level lcm426xx register access.

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

Include dependency graph for lcm426xxTransport.h:



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



## Classes

- struct [inv\\_icm426xx\\_serif](#)  
*basesensor serial interface*
- struct [inv\\_icm426xx\\_transport](#)  
*transport interface*
- struct [inv\\_icm426xx\\_transport::register\\_cache](#)  
*Contains mirrored values of some IP registers.*

## Enumerations

## Functions

- int [inv\\_icm426xx\\_init\\_transport](#) (struct [inv\\_icm426xx](#) \*s)  
*Init cache variable.*
- int [inv\\_icm426xx\\_read\\_reg](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t reg, uint32\_t len, uint8\_t \*buf)  
*Reads data from a register on lcm426xx.*
- int [inv\\_icm426xx\\_write\\_reg](#) (struct [inv\\_icm426xx](#) \*s, uint8\_t reg, uint32\_t len, const uint8\_t \*buf)  
*Writes data to a register on lcm426xx.*

### 6.6.1 Detailed Description

Low-level lcm426xx register access.



# Index

fifo\_header\_t, [39](#)

fifo\_is\_used

inv\_icm426xx, [41](#)

ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_16g

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_2g

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_4g

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_8g

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_t

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_100\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_12\_5\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_16\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_5625\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_200\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_25\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_2\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_32\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_3\_125\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_4\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_500\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_50\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_6\_25\_HZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_8\_KHZ

lcm426xxDefs.h, [50](#)

ICM426XX\_ACCEL\_CONFIG0\_ODR\_t

lcm426xxDefs.h, [50](#)

ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_25Hz

lcm426xxDefs.h, [51](#)

ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_50Hz

lcm426xxDefs.h, [51](#)

ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_RESERVED

lcm426xxDefs.h, [51](#)

ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_t

lcm426xxDefs.h, [50](#)

ICM426XX\_FIFO\_MIRRORING\_SIZE

lcm426xx driver high level functions, [10](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_1000dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_125dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_16dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_2000dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_250dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_31dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_500dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_62dps

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_t

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_100\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_12\_5\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_16\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_1\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_200\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_25\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_2\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_32\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_4\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_500\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_50\_HZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_8\_KHZ

lcm426xxDefs.h, [51](#)

ICM426XX\_GYRO\_CONFIG0\_ODR\_t

- lcm426xxDefs.h, [51](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_1
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_16
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_2
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_32
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_4
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_8
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_t
  - lcm426xxDefs.h, [51](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_16g
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_2g
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_4g
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_8g
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_t
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_↔
  - 1000dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_125dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_16dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_↔
  - 2000dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_250dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_31dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_500dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_62dps
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_t
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_1
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_16
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_2
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_32
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_4
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_8
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_t
  - lcm426xxDefs.h, [52](#)
- lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_16g
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_2g
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_4g
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_8g
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_t
  - lcm426xxDefs.h, [52](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_↔
  - 1000dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_125dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_16dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_↔
  - 2000dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_250dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_31dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_500dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_62dps
  - lcm426xxDefs.h, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_t
  - lcm426xxDefs.h, [53](#)
- INV\_ICM426XX\_FIFO\_CONFIG\_t
  - lcm426xx driver high level functions, [10](#)
- INV\_ICM426XX\_FIFO\_DISABLED
  - lcm426xx driver high level functions, [10](#)
- INV\_ICM426XX\_FIFO\_ENABLED
  - lcm426xx driver high level functions, [10](#)
- INV\_ICM426XX\_LIGHTWEIGHT\_DRIVER
  - lcm426xx driver high level functions, [10](#)
- INV\_ICM426XX\_SENSOR\_ACCEL
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER\_CO↔
  - UNT
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER\_EV↔
  - ENT
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_DMP\_R2W
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_DMP\_TILT
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_FSYNC\_EVENT
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_GYRO
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_OIS
  - lcm426xx driver high level functions, [11](#)
- INV\_ICM426XX\_SENSOR\_TAP
  - lcm426xx driver high level functions, [11](#)



- lcm426xx driver high level functions, 11
- INV\_ICM426XX\_SENSOR\_TEMPERATURE
  - lcm426xx driver high level functions, 11
- lcm426xx driver extern functions, 34
  - inv\_icm426xx\_get\_time\_us, 34
  - inv\_icm426xx\_sleep\_us, 34
- lcm426xx driver high level functions, 7
  - ICM426XX\_FIFO\_MIRRORING\_SIZE, 10
  - INV\_ICM426XX\_FIFO\_CONFIG\_t, 10
  - INV\_ICM426XX\_FIFO\_DISABLED, 10
  - INV\_ICM426XX\_FIFO\_ENABLED, 10
  - INV\_ICM426XX\_LIGHTWEIGHT\_DRIVER, 10
  - INV\_ICM426XX\_SENSOR\_ACCEL, 11
  - INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER↔\_COUNT, 11
  - INV\_ICM426XX\_SENSOR\_DMP\_PEDOMETER↔\_EVENT, 11
  - INV\_ICM426XX\_SENSOR\_DMP\_R2W, 11
  - INV\_ICM426XX\_SENSOR\_DMP\_TILT, 11
  - INV\_ICM426XX\_SENSOR\_FSYNC\_EVENT, 11
  - INV\_ICM426XX\_SENSOR\_GYRO, 11
  - INV\_ICM426XX\_SENSOR\_OIS, 11
  - INV\_ICM426XX\_SENSOR\_TAP, 11
  - INV\_ICM426XX\_SENSOR\_TEMPERATURE, 11
  - inv\_icm426xx\_configure\_fifo, 11
  - inv\_icm426xx\_configure\_fifo\_wm, 11
  - inv\_icm426xx\_configure\_timestamp\_resolution, 11
  - inv\_icm426xx\_convert\_odr\_bitfield\_to\_us, 12
  - inv\_icm426xx\_device\_reset, 12
  - inv\_icm426xx\_disable\_accel, 12
  - inv\_icm426xx\_disable\_fsync, 12
  - inv\_icm426xx\_disable\_gyro, 13
  - inv\_icm426xx\_disable\_high\_resolution\_fifo, 13
  - inv\_icm426xx\_disable\_timestamp\_to\_register, 13
  - inv\_icm426xx\_enable\_accel\_low\_noise\_mode, 13
  - inv\_icm426xx\_enable\_accel\_low\_power\_mode, 14
  - inv\_icm426xx\_enable\_clkin\_rtc, 14
  - inv\_icm426xx\_enable\_fsync, 14
  - inv\_icm426xx\_enable\_gyro\_low\_noise\_mode, 15
  - inv\_icm426xx\_enable\_high\_resolution\_fifo, 15
  - inv\_icm426xx\_enable\_timestamp\_to\_register, 15
  - inv\_icm426xx\_force\_clock\_source, 15
  - inv\_icm426xx\_get\_accel\_fsr, 16
  - inv\_icm426xx\_get\_clkin\_rtc\_status, 16
  - inv\_icm426xx\_get\_config\_ibi, 16
  - inv\_icm426xx\_get\_config\_int1, 17
  - inv\_icm426xx\_get\_config\_int2, 17
  - inv\_icm426xx\_get\_current\_timestamp, 17
  - inv\_icm426xx\_get\_data\_from\_fifo, 18
  - inv\_icm426xx\_get\_data\_from\_registers, 18
  - inv\_icm426xx\_get\_fifo\_timestamp\_resolution\_us↔\_q24, 18
  - inv\_icm426xx\_get\_gyro\_fsr, 18
  - inv\_icm426xx\_get\_reg\_timestamp\_resolution↔\_us\_q24, 19
  - inv\_icm426xx\_get\_version, 19
  - inv\_icm426xx\_get\_who\_am\_i, 19
  - inv\_icm426xx\_init, 19
  - inv\_icm426xx\_reset\_fifo, 20
  - inv\_icm426xx\_sensor, 10
  - inv\_icm426xx\_set\_accel\_frequency, 20
  - inv\_icm426xx\_set\_accel\_fsr, 21
  - inv\_icm426xx\_set\_accel\_ln\_bw, 21
  - inv\_icm426xx\_set\_accel\_lp\_avg, 22
  - inv\_icm426xx\_set\_config\_ibi, 22
  - inv\_icm426xx\_set\_config\_int1, 22
  - inv\_icm426xx\_set\_config\_int2, 23
  - inv\_icm426xx\_set\_gyro\_frequency, 23
  - inv\_icm426xx\_set\_gyro\_fsr, 23
  - inv\_icm426xx\_set\_gyro\_ln\_bw, 24
  - inv\_icm426xx\_set\_reg\_bank, 24
- lcm426xx driver high level functions related to APEX, 25
  - inv\_icm426xx\_configure\_apex\_parameters, 26
  - inv\_icm426xx\_configure\_smd\_wom, 27
  - inv\_icm426xx\_configure\_tap\_parameters, 27
  - inv\_icm426xx\_disable\_apex\_pedometer, 28
  - inv\_icm426xx\_disable\_apex\_r2w, 28
  - inv\_icm426xx\_disable\_apex\_tilt, 28
  - inv\_icm426xx\_disable\_smd, 28
  - inv\_icm426xx\_disable\_tap, 28
  - inv\_icm426xx\_disable\_wom, 29
  - inv\_icm426xx\_enable\_apex\_pedometer, 29
  - inv\_icm426xx\_enable\_apex\_r2w, 29
  - inv\_icm426xx\_enable\_apex\_tilt, 29
  - inv\_icm426xx\_enable\_smd, 30
  - inv\_icm426xx\_enable\_tap, 30
  - inv\_icm426xx\_enable\_wom, 30
  - inv\_icm426xx\_get\_apex\_data\_activity, 30
  - inv\_icm426xx\_get\_apex\_parameters, 31
  - inv\_icm426xx\_get\_tap\_data, 31
  - inv\_icm426xx\_get\_tap\_parameters, 31
  - inv\_icm426xx\_init\_apex\_parameters\_struct, 32
  - inv\_icm426xx\_init\_tap\_parameters\_struct, 32
  - inv\_icm426xx\_set\_apex\_frequency, 32
  - inv\_icm426xx\_start\_dmp, 33
- lcm426xx driver transport, 37
  - inv\_icm426xx\_init\_transport, 37
  - inv\_icm426xx\_read\_reg, 37
  - inv\_icm426xx\_write\_reg, 38
- lcm426xx selftest, 35
  - inv\_icm426xx\_get\_st\_bias, 35
  - inv\_icm426xx\_run\_selftest, 35
  - inv\_icm426xx\_set\_st\_bias, 36
- lcm426xxDefs.h, 49
  - ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_16g, 50
  - ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_2g, 50
  - ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_4g, 50
  - ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_8g, 50
  - ICM426XX\_ACCEL\_CONFIG0\_FS\_SEL\_t, 50
  - ICM426XX\_ACCEL\_CONFIG0\_ODR\_100\_HZ, 50
  - ICM426XX\_ACCEL\_CONFIG0\_ODR\_12\_5\_HZ, 50
  - ICM426XX\_ACCEL\_CONFIG0\_ODR\_16\_KHZ, 50
  - ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_5625↔\_HZ, 50
  - ICM426XX\_ACCEL\_CONFIG0\_ODR\_1\_KHZ, 50

- ICM426XX\_ACCEL\_CONFIG0\_ODR\_200\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_25\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_2\_KHZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_32\_KHZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_3\_125\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_4\_KHZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_500\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_50\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_6\_25\_HZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_8\_KHZ, [50](#)
- ICM426XX\_ACCEL\_CONFIG0\_ODR\_t, [50](#)
- ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_25Hz, [51](#)
- ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_50Hz, [51](#)
- ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_RESERVED, [51](#)
- ICM426XX\_APEX\_CONFIG0\_DMP\_ODR\_t, [50](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_1000dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_125dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_16dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_2000dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_250dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_31dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_500dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_62dps, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_FS\_SEL\_t, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_100\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_12\_5\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_16\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_1\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_200\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_25\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_2\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_32\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_4\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_500\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_50\_HZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_8\_KHZ, [51](#)
- ICM426XX\_GYRO\_CONFIG0\_ODR\_t, [51](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_1, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_16, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_2, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_32, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_4, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_8, [52](#)
- ICM426XX\_OIS1\_CONFIG1\_DEC\_t, [51](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_16g, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_2g, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_4g, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_8g, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_ACCEL\_FS\_SEL\_t, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_1000dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_125dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_16dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_2000dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_250dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_31dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_500dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_62dps, [52](#)
- ICM426XX\_OIS1\_CONFIG2\_GYRO\_FS\_SEL\_t, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_1, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_16, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_2, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_32, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_4, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_8, [52](#)
- ICM426XX\_OIS2\_CONFIG1\_DEC\_t, [52](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_16g, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_2g, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_4g, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_8g, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_ACCEL\_FS\_SEL\_t, [52](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_1000dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_125dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_16dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_2000dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_250dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_31dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_500dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_62dps, [53](#)
- ICM426XX\_OIS2\_CONFIG2\_GYRO\_FS\_SEL\_t, [53](#)

- 53
- lcm426xxDriver\_HL.h, 53
- lcm426xxDriver\_HL\_apex.h, 57
- lcm426xxExtFunc.h, 59
- lcm426xxSelfTest.h, 60
- lcm426xxTransport.h, 60
- inv\_icm426xx, 39
  - fifo\_is\_used, 41
  - sensor\_event\_cb, 41
- inv\_icm426xx\_apex\_parameters, 41
  - power\_save, 42
  - r2w\_gest\_delay, 42
  - r2w\_mounting\_matrix, 42
  - r2w\_sleep\_time\_out, 43
  - sensitivity\_mode, 43
- inv\_icm426xx\_apex\_step\_activity, 43
  - step\_cadence, 43
- inv\_icm426xx\_configure\_apex\_parameters
  - lcm426xx driver high level functions related to A↔ PEX, 26
- inv\_icm426xx\_configure\_fifo
  - lcm426xx driver high level functions, 11
- inv\_icm426xx\_configure\_fifo\_wm
  - lcm426xx driver high level functions, 11
- inv\_icm426xx\_configure\_smd\_wom
  - lcm426xx driver high level functions related to A↔ PEX, 27
- inv\_icm426xx\_configure\_tap\_parameters
  - lcm426xx driver high level functions related to A↔ PEX, 27
- inv\_icm426xx\_configure\_timestamp\_resolution
  - lcm426xx driver high level functions, 11
- inv\_icm426xx\_convert\_odr\_bitfield\_to\_us
  - lcm426xx driver high level functions, 12
- inv\_icm426xx\_device\_reset
  - lcm426xx driver high level functions, 12
- inv\_icm426xx\_disable\_accel
  - lcm426xx driver high level functions, 12
- inv\_icm426xx\_disable\_apex\_pedometer
  - lcm426xx driver high level functions related to A↔ PEX, 28
- inv\_icm426xx\_disable\_apex\_r2w
  - lcm426xx driver high level functions related to A↔ PEX, 28
- inv\_icm426xx\_disable\_apex\_tilt
  - lcm426xx driver high level functions related to A↔ PEX, 28
- inv\_icm426xx\_disable\_fsync
  - lcm426xx driver high level functions, 12
- inv\_icm426xx\_disable\_gyro
  - lcm426xx driver high level functions, 13
- inv\_icm426xx\_disable\_high\_resolution\_fifo
  - lcm426xx driver high level functions, 13
- inv\_icm426xx\_disable\_smd
  - lcm426xx driver high level functions related to A↔ PEX, 28
- inv\_icm426xx\_disable\_tap
  - lcm426xx driver high level functions related to A↔ PEX, 28
- inv\_icm426xx\_disable\_timestamp\_to\_register
  - lcm426xx driver high level functions, 13
- inv\_icm426xx\_disable\_wom
  - lcm426xx driver high level functions related to A↔ PEX, 29
- inv\_icm426xx\_enable\_accel\_low\_noise\_mode
  - lcm426xx driver high level functions, 13
- inv\_icm426xx\_enable\_accel\_low\_power\_mode
  - lcm426xx driver high level functions, 14
- inv\_icm426xx\_enable\_apex\_pedometer
  - lcm426xx driver high level functions related to A↔ PEX, 29
- inv\_icm426xx\_enable\_apex\_r2w
  - lcm426xx driver high level functions related to A↔ PEX, 29
- inv\_icm426xx\_enable\_apex\_tilt
  - lcm426xx driver high level functions related to A↔ PEX, 29
- inv\_icm426xx\_enable\_clkln\_rtc
  - lcm426xx driver high level functions, 14
- inv\_icm426xx\_enable\_fsync
  - lcm426xx driver high level functions, 14
- inv\_icm426xx\_enable\_gyro\_low\_noise\_mode
  - lcm426xx driver high level functions, 15
- inv\_icm426xx\_enable\_high\_resolution\_fifo
  - lcm426xx driver high level functions, 15
- inv\_icm426xx\_enable\_smd
  - lcm426xx driver high level functions related to A↔ PEX, 30
- inv\_icm426xx\_enable\_tap
  - lcm426xx driver high level functions related to A↔ PEX, 30
- inv\_icm426xx\_enable\_timestamp\_to\_register
  - lcm426xx driver high level functions, 15
- inv\_icm426xx\_enable\_wom
  - lcm426xx driver high level functions related to A↔ PEX, 30
- inv\_icm426xx\_force\_clock\_source
  - lcm426xx driver high level functions, 15
- inv\_icm426xx\_get\_accel\_fsr
  - lcm426xx driver high level functions, 16
- inv\_icm426xx\_get\_apex\_data\_activity
  - lcm426xx driver high level functions related to A↔ PEX, 30
- inv\_icm426xx\_get\_apex\_parameters
  - lcm426xx driver high level functions related to A↔ PEX, 31
- inv\_icm426xx\_get\_clkln\_rtc\_status
  - lcm426xx driver high level functions, 16
- inv\_icm426xx\_get\_config\_ibi
  - lcm426xx driver high level functions, 16
- inv\_icm426xx\_get\_config\_int1
  - lcm426xx driver high level functions, 17
- inv\_icm426xx\_get\_config\_int2
  - lcm426xx driver high level functions, 17
- inv\_icm426xx\_get\_current\_timestamp

- lcm426xx driver high level functions, [17](#)
- inv\_icm426xx\_get\_data\_from\_fifo
  - lcm426xx driver high level functions, [18](#)
- inv\_icm426xx\_get\_data\_from\_registers
  - lcm426xx driver high level functions, [18](#)
- inv\_icm426xx\_get\_fifo\_timestamp\_resolution\_us\_q24
  - lcm426xx driver high level functions, [18](#)
- inv\_icm426xx\_get\_gyro\_fsr
  - lcm426xx driver high level functions, [18](#)
- inv\_icm426xx\_get\_reg\_timestamp\_resolution\_us\_q24
  - lcm426xx driver high level functions, [19](#)
- inv\_icm426xx\_get\_st\_bias
  - lcm426xx selftest, [35](#)
- inv\_icm426xx\_get\_tap\_data
  - lcm426xx driver high level functions related to A↔PEX, [31](#)
- inv\_icm426xx\_get\_tap\_parameters
  - lcm426xx driver high level functions related to A↔PEX, [31](#)
- inv\_icm426xx\_get\_time\_us
  - lcm426xx driver extern functions, [34](#)
- inv\_icm426xx\_get\_version
  - lcm426xx driver high level functions, [19](#)
- inv\_icm426xx\_get\_who\_am\_i
  - lcm426xx driver high level functions, [19](#)
- inv\_icm426xx\_init
  - lcm426xx driver high level functions, [19](#)
- inv\_icm426xx\_init\_apex\_parameters\_struct
  - lcm426xx driver high level functions related to A↔PEX, [32](#)
- inv\_icm426xx\_init\_tap\_parameters\_struct
  - lcm426xx driver high level functions related to A↔PEX, [32](#)
- inv\_icm426xx\_init\_transport
  - lcm426xx driver transport, [37](#)
- inv\_icm426xx\_interrupt\_parameter\_t, [44](#)
- inv\_icm426xx\_read\_reg
  - lcm426xx driver transport, [37](#)
- inv\_icm426xx\_reset\_fifo
  - lcm426xx driver high level functions, [20](#)
- inv\_icm426xx\_run\_selftest
  - lcm426xx selftest, [35](#)
- inv\_icm426xx\_sensor
  - lcm426xx driver high level functions, [10](#)
- inv\_icm426xx\_sensor\_event\_t, [44](#)
- inv\_icm426xx\_serif, [44](#)
- inv\_icm426xx\_set\_accel\_frequency
  - lcm426xx driver high level functions, [20](#)
- inv\_icm426xx\_set\_accel\_fsr
  - lcm426xx driver high level functions, [21](#)
- inv\_icm426xx\_set\_accel\_ln\_bw
  - lcm426xx driver high level functions, [21](#)
- inv\_icm426xx\_set\_accel\_lp\_avg
  - lcm426xx driver high level functions, [22](#)
- inv\_icm426xx\_set\_apex\_frequency
  - lcm426xx driver high level functions related to A↔PEX, [32](#)
- inv\_icm426xx\_set\_config\_ibi
  - lcm426xx driver high level functions, [22](#)
- inv\_icm426xx\_set\_config\_int1
  - lcm426xx driver high level functions, [22](#)
- inv\_icm426xx\_set\_config\_int2
  - lcm426xx driver high level functions, [23](#)
- inv\_icm426xx\_set\_gyro\_frequency
  - lcm426xx driver high level functions, [23](#)
- inv\_icm426xx\_set\_gyro\_fsr
  - lcm426xx driver high level functions, [23](#)
- inv\_icm426xx\_set\_gyro\_ln\_bw
  - lcm426xx driver high level functions, [24](#)
- inv\_icm426xx\_set\_reg\_bank
  - lcm426xx driver high level functions, [24](#)
- inv\_icm426xx\_set\_st\_bias
  - lcm426xx selftest, [36](#)
- inv\_icm426xx\_sleep\_us
  - lcm426xx driver extern functions, [34](#)
- inv\_icm426xx\_start\_dmp
  - lcm426xx driver high level functions related to A↔PEX, [33](#)
- inv\_icm426xx\_tap\_data, [45](#)
- inv\_icm426xx\_tap\_parameters\_t, [45](#)
- inv\_icm426xx\_transport, [46](#)
  - register\_cache, [47](#)
- inv\_icm426xx\_transport::register\_cache, [47](#)
- inv\_icm426xx\_write\_reg
  - lcm426xx driver transport, [38](#)
- power\_save
  - inv\_icm426xx\_apex\_parameters, [42](#)
- r2w\_gest\_delay
  - inv\_icm426xx\_apex\_parameters, [42](#)
- r2w\_mounting\_matrix
  - inv\_icm426xx\_apex\_parameters, [42](#)
- r2w\_sleep\_time\_out
  - inv\_icm426xx\_apex\_parameters, [43](#)
- recover\_regs, [47](#)
- register\_cache
  - inv\_icm426xx\_transport, [47](#)
- sensitivity\_mode
  - inv\_icm426xx\_apex\_parameters, [43](#)
- sensor\_event\_cb
  - inv\_icm426xx, [41](#)
- step\_cadence
  - inv\_icm426xx\_apex\_step\_activity, [43](#)