IIS2ICLX: high-accuracy, high-resolution, low-power, 2-axis digital inclinometer

Introduction

This document is intended to provide usage information and application hints related to ST's IIS2ICLX inclinometer.

The IIS2ICLX has a selectable full scale of ±0.5/±1/±2/±3 *g* and is capable of providing the measured accelerations to the application over an I²C or SPI digital interface.

Its high accuracy, stability over temperature and repeatability make IIS2ICLX particularly suitable for inclination measurement applications (inclinometers).

The sensing element is manufactured using a dedicated micromachining process developed by STMicroelectronics to produce inertial sensors and actuators on silicon wafers.

The IC interface is manufactured using a CMOS process that allows a high level of integration to design a dedicated circuit which is trimmed to better match the characteristics of the sensing element.

The IIS2ICLX has an unmatched set of embedded features (programmable FSM, Machine Learning Core, sensor hub, FIFO, event decoding and interrupts) which are enablers for implementing smart and complex sensor nodes which deliver high accuracy and performance at very low power.

The IIS2ICLX is available in a high-performance (low-stress) ceramic cavity land grid array (CC LGA) package and can operate within a temperature range of -40 °C to +105 °C.

1 Pin description

Figure 1. Pin connections

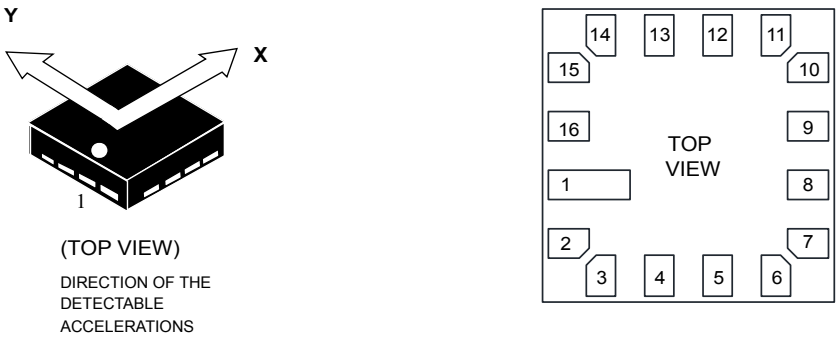


Table 1. Pin status

Pin #	Name	Mode 1 function	Mode 2 function	Pin status Mode 1	Pin status Mode 2
1	VDD_IO	Power supply for I/O pins (recommended 100 nF filter capacitor)			
2	CS	I²C/SPI mode selection (1:SPI idle mode / I²C communication enabled; 0: SPI communication mode / I²C disabled)		Default: input with pull-up. Pull-up is disabled if bit I2C_disable = 1 in reg 13h and DEVICE_CONF = 1 in reg 18h	Default: input with pull-up. Pull-up is disabled if bit I2C_disable = 1 in reg 13h and DEVICE_CONF = 1 in reg 18
3	GND	0 V supply			
4	INT2(1)	Programmable interrupt 2 (INT2) / Data enable (DEN)	Programmable interrupt 2 (INT2) / Data enable (DEN) / I²C master external synchronization signal (MDRDY)	Default: output forced to ground	Default: output forced to ground
5	SDO	SPI 4-wire interface serial data output (SDO)		Default: input without pull-up. Pull-up is enabled if bit SDO_PU_EN = 1 in reg. 02h.	Default: input without pull-up. Pull-up is enabled if bit SDO_PU_EN = 1 in reg. 02h.
	SA0	I²C least significant bit of the device address (SA0)			
6	INT1(2)	Programmable Interrupt 1 (INT1)		Default: input with pull-down	Default: input with pull-down
7	SDx	Connect to GND or VDD_IO	I²C serial data master (MSDA)	Default: input without pull-up. Pull-up is enabled if bit SHUB_PU_EN = 1 in reg 14h in sensor hub registers.	Default: input without pull-up. Pull-up is enabled if bit SHUB_PU_EN = 1 in reg 14h in sensor hub registers.
8	VDD	Power supply (recommended 100 nF filter capacitor)			
9	VDD	Power supply (recommended 100 nF filter capacitor)			
10	SCL	I²C serial clock (SCL) / SPI serial port clock (SPC)		Default: input without pull-up	Default: input without pull-up
11	SDA/SDI	I²C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)		Default: input without pull-up	Default: input without pull-up
12	SCx	Connect to GND or VDD_IO	I²C serial clock master (MSCL)	Default: input without pull-up. Pull-up is enabled if bit SHUB_PU_EN = 1 in reg 14h in sensor hub registers.	Default: input without pull-up. Pull-up is enabled if bit SHUB_PU_EN = 1 in reg 14h in sensor hub registers.
13	GND	GND			
14	NC	Connect to GND or leave unconnected			
15	NC	Connect to GND or leave unconnected			
16	NC	Connect to GND or leave unconnected			

1. *If no interrupt signal is needed on INT2, this pin can be left unconnected.*
2. *INT1 must be set to '0' or left unconnected during power-on. If no interrupt signal is needed on INT1, this pin can be left unconnected.*

Internal pull-up value is from 30 kΩ to 50 kΩ, depending on VDDIO.

Note: The procedure to correctly initialize the device is as follows:

1. *INT1 pin: leave unconnected or connect with external pull-down during power-on. Pull-up must be avoided on this pin.*
2. *INT2 pin: not recommended to connect with external pull-up.*
3. *Properly configure the device:*
   a. *SPI case: I2C_disable = 1 in CTRL4_C (13h) and DEVICE_CONF = 1 in CTRL9_XL (18h).*
   b. *I²C case: I2C_disable = 0 (default) in CTRL4_C (13h) and DEVICE_CONF = 1 in CTRL9_XL (18h).*

2 Registers

Table 2. Registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FUNC_CFG_ACCESS	01h	FUNC_CFG _ACCESS	SHUB_REG _ACCESS	0	0	0	0	0	0
PIN_CTRL	02h	0	SDO_PU_EN	1	1	1	1	1	1
FIFO_CTRL1	07h	WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0
FIFO_CTRL2	08h	STOP_ON_WTM	0	0	ODRCHG_EN	0	0	0	WTM8
FIFO_CTRL3	09h	0	0	0	0	BDR_XL_3	BDR_XL_2	BDR_XL_1	BDR_XL_0
FIFO_CTRL4	0Ah	DEC_TS_BATCH_1	DEC_TS_BATCH_0	ODR_T_BATCH_1	ODR_T_BATCH_0	0	FIFO_MODE2	FIFO_MODE1	FIFO_MODE0
COUNTER_BDR_REG1	0Bh	dataready_ pulsed	RST_COUNTER _BDR	0	0	0	0	0	CNT_BDR_TH_8
COUNTER_BDR_REG2	0Ch	CNT_BDR_TH_7	CNT_BDR_TH_6	CNT_BDR_TH_5	CNT_BDR_TH_4	CNT_BDR_TH_3	CNT_BDR_TH_2	CNT_BDR_TH_1	CNT_BDR_TH_0
INT1_CTRL	0Dh	DEN_DRDY_flag	INT1_CNT_BDR	INT1_FIFO_FULL	INT1_FIFO_OVR	INT1_FIFO_TH	INT1_BOOT	0	INT1_DRDY _XL
INT2_CTRL	0Eh	0	INT2_CNT_BDR	INT2_FIFO_FULL	INT2_FIFO_OVR	INT2_FIFO_TH	INT2_DRDY _TEMP	0	INT2_DRDY _XL
WHO_AM_I	0Fh	0	1	1	0	1	0	1	1
CTRL1_XL	10h	ODR_XL3	ODR_XL2	ODR_XL1	ODR_XL0	FS1_XL	FS0_XL	LPF2_XL_EN	0
CTRL3_C	12h	BOOT	BDU	H_LACTIVE	PP_OD	SIM	IF_INC	0	SW_RESET
CTRL4_C	13h	0	0	INT2_on_INT1	0	DRDY _MASK	I2C_disable	0	0
CTRL5_C	14h	0	0	0	0	0	0	ST1_XL	ST0_XL
CTRL6_C	15h	TRIG_EN	LVL1_EN	LVL2_EN	0	USR_OFF_W	0	0	0
CTRL7_XL	16h	0	0	0	0	0	0	USR_OFF_ ON_OUT	0
CTRL8_XL	17h	HPCF_XL2	HPCF_XL1	HPCF_XL0	HP_REF _MODE_XL	FASTSETTL _MODE_XL	HP_SLOPE_XL_EN	0	0
CTRL9_XL	18h	DEN_X	DEN_Y	1	1	DEN_EN	DEN_LH	DEVICE_CONF	0
CTRL10_C	19h	0	0	TIMESTAMP_EN	0	0	0	0	0
ALL_INT_SRC	1Ah	TIMESTAMP _ENDCOUNT	0	SLEEP_ CHANGE_IA	0	DOUBLE_TAP	SINGLE_TAP	WU_IA	0
WAKE_UP_SRC	1Bh	0	SLEEP_ CHANGE_IA	0	SLEEP_STATE	WU_IA	X_WU	Y_WU	0
TAP_SRC	1Ch	0	TAP_IA	SINGLE_TAP	DOUBLE_TAP	TAP_SIGN	X_TAP	Y_TAP	0
DEN_SRC	1Dh	DEN_DRDY	0	0	0	0	0	0	0
STATUS_REG	1Eh	0	0	0	0	0	TDA	0	XLDA
OUT_TEMP_L	20h	Temp7	Temp6	Temp5	Temp4	Temp3	Temp2	Temp1	Temp0
OUT_TEMP_H	21h	Temp15	Temp14	Temp13	Temp12	Temp11	Temp10	Temp9	Temp8
OUTX_L_A	28h	D7	D6	D5	D4	D3	D2	D1	D0

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OUTX_H_A	29h	D15	D14	D13	D12	D11	D10	D9	D8
OUTY_L_A	2Ah	D7	D6	D5	D4	D3	D2	D1	D0
OUTY_H_A	2Bh	D15	D14	D13	D12	D11	D10	D9	D8
EMB_FUNC_ STATUS_MAINPAGE	35h	IS_FSM_LC	0	0	0	0	0	0	0
FSM_STATUS_A _MAINPAGE	36h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
FSM_STATUS_B _MAINPAGE	37h	IS_FSM16	IS_FSM15	IS_FSM14	IS_FSM13	IS_FSM12	IS_FSM11	IS_FSM10	IS_FSM9
MLC_STATUS _MAINPAGE	38h	IS_MLC8	IS_MLC7	IS_MLC6	IS_MLC5	IS_MLC4	IS_MLC3	IS_MLC2	IS_MLC1
STATUS_MASTER _MAINPAGE	39h	WR_ONCE_DONE	SLAVE3_NACK	SLAVE2_NACK	SLAVE1_NACK	SLAVE0_NACK	0	0	SENS_HUB _ENDOP
FIFO_STATUS1	3Ah	DIFF_FIFO_7	DIFF_FIFO_6	DIFF_FIFO_5	DIFF_FIFO_4	DIFF_FIFO_3	DIFF_FIFO_2	DIFF_FIFO_1	DIFF_FIFO_0
FIFO_STATUS2	3Bh	FIFO_WTM_IA	FIFO_OVR_IA	FIFO_FULL_IA	COUNTER _BDR_IA	FIFO_OVR _LATCHED	0	DIFF_FIFO_9	DIFF_FIFO_8
TIMESTAMP0	40h	T7	T6	T5	T4	T3	T2	T1	T0
TIMESTAMP1	41h	T15	T14	T13	T12	T11	T10	T9	T8
TIMESTAMP2	42h	T23	T22	T21	T20	T19	T18	T17	T16
TIMESTAMP3	43h	T31	T30	T29	T28	T27	T26	T25	T24
TAP_CFG0	56h	0	INT_CLR_ ON_READ	SLEEP_STATUS _ON_INT	SLOPE_FDS	TAP_X_EN	TAP_Y_EN	0	LIR
TAP_CFG1	57h	0	0	TAP_PRIORITY	TAP_THS_X_4	TAP_THS_X_3	TAP_THS_X_2	TAP_THS_X_1	TAP_THS_X_0
TAP_CFG2	58h	INTERRUPTS _ENABLE	0	0	TAP_THS_Y_4	TAP_THS_Y_3	TAP_THS_Y_2	TAP_THS_Y_1	TAP_THS_Y_0
INT_DUR2	5Ah	DUR3	DUR2	DUR1	DUR0	QUIET1	QUIET0	SHOCK1	SHOCK0
WAKE_UP_THS	5Bh	SINGLE_ DOUBLE_TAP	USR_OFF_ON_WU	WK_THS5	WK_THS4	WK_THS3	WK_THS2	WK_THS1	WK_THS0
WAKE_UP_DUR	5Ch	0	WAKE_DUR1	WAKE_DUR0	WAKE_THS_W	SLEEP_DUR3	SLEEP_DUR2	SLEEP_DUR1	SLEEP_DUR0
MD1_CFG	5Eh	INT1_SLEEP _CHANGE	INT1_ SINGLE_TAP	INT1_WU	0	INT1_ DOUBLE_TAP	0	INT1_EMB_FUNC	INT1_SHUB
MD2_CFG	5Fh	INT2_SLEEP _CHANGE	INT2_ SINGLE_TAP	INT2_WU	0	INT2_ DOUBLE_TAP	0	INT2_EMB_FUNC	INT2_TIMESTAMP
INTERNAL_FREQ_FINE	63h	FREQ_FINE7	FREQ_FINE6	FREQ_FINE5	FREQ_FINE4	FREQ_FINE3	FREQ_FINE2	FREQ_FINE1	FREQ_FINE0
X_OFS_USR	73h	X_OFS_USR_7	X_OFS_USR_6	X_OFS_USR_5	X_OFS_USR_4	X_OFS_USR_3	X_OFS_USR_2	X_OFS_USR_1	X_OFS_USR_0
Y_OFS_USR	74h	Y_OFS_USR_7	Y_OFS_USR_6	Y_OFS_USR_5	Y_OFS_USR_4	Y_OFS_USR_3	Y_OFS_USR_2	Y_OFS_USR_1	Y_OFS_USR_0
FIFO_DATA_OUT_TAG	78h	TAG_SENSOR_4	TAG_SENSOR_3	TAG_SENSOR_2	TAG_SENSOR_1	TAG_SENSOR_0	TAG_CNT_1	TAG_CNT_0	TAG_PARITY
FIFO_DATA_OUT_X_L	79h	D7	D6	D5	D4	D3	D2	D1	D0
FIFO_DATA_OUT_X_H	7Ah	D15	D14	D13	D12	D11	D10	D9	D8
FIFO_DATA_OUT_Y_L	7Bh	D7	D6	D5	D4	D3	D2	D1	D0

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FIFO_DATA_OUT_Y_H	7Ch	D15	D14	D13	D12	D11	D10	D9	D8
FIFO_DATA_OUT_Z_L[1]	7Dh	D7	D6	D5	D4	D3	D2	D1	D0
FIFO_DATA_OUT_Z_H[1]	7Eh	D15	D14	D13	D12	D11	D10	D9	D8

1. Z-axis support is available for storing external sensor data in FIFO.

2.1 Embedded functions registers

The table given below provides a list of the registers for the embedded functions available in the device and the corresponding addresses. Embedded functions registers are accessible when FUNC_CFG_ACCESS is set to 1 in the FUNC_CFG_ACCESS register.

Table 3. Embedded functions registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAGE_SEL	02h	PAGE_SEL3	PAGE_SEL2	PAGE_SEL1	PAGE_SEL0	0	0	0	1
EMB_FUNC_EN_B	05h	0	0	0	MLC_EN	0	0	0	FSM_EN
PAGE_ADDRESS	08h	PAGE_ADDR7	PAGE_ADDR6	PAGE_ADDR5	PAGE_ADDR4	PAGE_ADDR3	PAGE_ADDR2	PAGE_ADDR1	PAGE_ADDR0
PAGE_VALUE	09h	PAGE_VALUE7	PAGE_VALUE6	PAGE_VALUE5	PAGE_VALUE4	PAGE_VALUE3	PAGE_VALUE2	PAGE_VALUE1	PAGE_VALUE0
EMB_FUNC_INT1	0Ah	INT1_FSM_LC	0	0	0	0	0	0	0
FSM_INT1_A	0Bh	INT1_FSM8	INT1_FSM7	INT1_FSM6	INT1_FSM5	INT1_FSM4	INT1_FSM3	INT1_FSM2	INT1_FSM1
FSM_INT1_B	0Ch	INT1_FSM16	INT1_FSM15	INT1_FSM14	INT1_FSM13	INT1_FSM12	INT1_FSM11	INT1_FSM10	INT1_FSM9
MLC_INT1	0Dh	INT1_MLC8	INT1_MLC7	INT1_MLC6	INT1_MLC5	INT1_MLC4	INT1_MLC3	INT1_MLC2	INT1_MLC1
EMB_FUNC_INT2	0Eh	INT2_FSM_LC	0	0	0	0	0	0	0
FSM_INT2_A	0Fh	INT2_FSM8	INT2_FSM7	INT2_FSM6	INT2_FSM5	INT2_FSM4	INT2_FSM3	INT2_FSM2	INT2_FSM1
FSM_INT2_B	10h	INT2_FSM16	INT2_FSM15	INT2_FSM14	INT2_FSM13	INT2_FSM12	INT2_FSM11	INT2_FSM10	INT2_FSM9
MLC_INT2	11h	INT2_MLC8	INT2_MLC7	INT2_MLC6	INT2_MLC6	INT2_MLC4	INT2_MLC3	INT2_MLC2	INT2_MLC1
EMB_FUNC_STATUS	12h	IS_FSM_LC	0	0	0	0	0	0	0
FSM_STATUS_A	13h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
FSM_STATUS_B	14h	IS_FSM16	IS_FSM15	IS_FSM14	IS_FSM13	IS_FSM12	IS_FSM11	IS_FSM10	IS_FSM9
MLC_STATUS	15h	IS_MLC8	IS_MLC7	IS_MLC6	IS_MLC5	IS_MLC4	IS_MLC3	IS_MLC2	IS_MLC1
PAGE_RW	17h	EMB_FUNC_LIR	PAGE_WRITE	PAGE_READ	0	0	0	0	0
FSM_ENABLE_A	46h	FSM8_EN	FSM7_EN	FSM6_EN	FSM5_EN	FSM4_EN	FSM3_EN	FSM2_EN	FSM1_EN
FSM_ENABLE_B	47h	FSM16_EN	FSM15_EN	FSM14_EN	FSM13_EN	FSM12_EN	FSM11_EN	FSM10_EN	FSM9_EN
FSM_LONG_COUNTER_L	48h	FSM_LC_7	FSM_LC_6	FSM_LC_5	FSM_LC_4	FSM_LC_3	FSM_LC_2	FSM_LC_1	FSM_LC_0
FSM_LONG_COUNTER_H	49h	FSM_LC_15	FSM_LC_14	FSM_LC_13	FSM_LC_12	FSM_LC_11	FSM_LC_10	FSM_LC_9	FSM_LC_8
FSM_LONG_ COUNTER_CLEAR	4Ah	0	0	0	0	0	0	FSM_LC_ CLEARED	FSM_LC_ CLEAR
FSM_OUTS1	4Ch	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS2	4Dh	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS3	4Eh	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS4	4Fh	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS5	50h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS6	51h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_OUTS7	52h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS8	53h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS9	54h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS10	55h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS11	56h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS12	57h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS13	58h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS14	59h	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS15	5Ah	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
FSM_OUTS16	5Bh	P_X	N_X	P_Y	N_Y	P_Z(1)	N_Z(1)	0	0
EMB_FUNC_ODR_CFG_B	5Fh	0	1	0	FSM_ODR1	FSM_ODR0	0	1	1
EMB_FUNC_ODR_CFG_C	60h	0	0	MLC_ODR1	MLC_ODR0	0	1	0	1
EMB_FUNC_INIT_B	67h	0	0	0	MLC_INIT	0	0	0	FSM_INIT
MLC0_SRC	70h	MLC0_SRC_7	MLC0_SRC_6	MLC0_SRC_5	MLC0_SRC_4	MLC0_SRC_3	MLC0_SRC_2	MLC0_SRC_1	MLC0_SRC_70
MLC1_SRC	71h	MLC1_SRC_7	MLC1_SRC_6	MLC1_SRC_5	MLC1_SRC_4	MLC1_SRC_3	MLC1_SRC_2	MLC1_SRC_1	MLC1_SRC_0
MLC2_SRC	72h	MLC2_SRC_7	MLC2_SRC_6	MLC2_SRC_5	MLC2_SRC_4	MLC2_SRC_3	MLC2_SRC_2	MLC2_SRC_1	MLC2_SRC_0
MLC3_SRC	73h	MLC3_SRC_7	MLC3_SRC_6	MLC3_SRC_5	MLC3_SRC_4	MLC3_SRC_3	MLC3_SRC_2	MLC3_SRC_1	MLC3_SRC_0
MLC4_SRC	74h	MLC4_SRC_7	MLC4_SRC_6	MLC4_SRC_5	MLC4_SRC_4	MLC4_SRC_3	MLC4_SRC_2	MLC4_SRC_1	MLC4_SRC_0
MLC5_SRC	75h	MLC5_SRC_7	MLC5_SRC_6	MLC5_SRC_5	MLC5_SRC_4	MLC5_SRC_3	MLC5_SRC_2	MLC5_SRC_1	MLC5_SRC_0
MLC6_SRC	76h	MLC6_SRC_7	MLC6_SRC_6	MLC6_SRC_5	MLC6_SRC_4	MLC6_SRC_3	MLC6_SRC_2	MLC6_SRC_1	MLC6_SRC_0
MLC7_SRC	77h	MLC7_SRC_7	MLC7_SRC_6	MLC7_SRC_5	MLC7_SRC_4	MLC7_SRC_3	MLC7_SRC_2	MLC7_SRC_1	MLC7_SRC_0

1. *Z-axis support is available for processing external sensor data.*

2.2 Embedded advanced features pages

The following table provides a list of the registers for the embedded advanced features page 1. These registers are accessible when PAGE_SEL[3:0] are set to 0001b in the PAGE_SEL register.

Table 4. Embedded advanced features registers - page 1

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC_TIMEOUT_L	7Ah	FSM_LC_ TIMEOUT7	FSM_LC_ TIMEOUT6	FSM_LC_ TIMEOUT5	FSM_LC_ TIMEOUT4	FSM_LC_ TIMEOUT3	FSM_LC_ TIMEOUT2	FSM_LC_ TIMEOUT1	FSM_LC_ TIMEOUT0
FSM_LC_TIMEOUT_H	7Bh	FSM_LC_ TIMEOUT15	FSM_LC_ TIMEOUT14	FSM_LC_ TIMEOUT13	FSM_LC_ TIMEOUT12	FSM_LC_ TIMEOUT11	FSM_LC_ TIMEOUT10	FSM_LC_ TIMEOUT9	FSM_LC_ TIMEOUT8
FSM_PROGRAMS	7Ch	FSM_N_PROG7	FSM_N_PROG6	FSM_N_PROG5	FSM_N_PROG4	FSM_N_PROG3	FSM_N_PROG2	FSM_N_PROG1	FSM_N_PROG0
FSM_START_ADD_L	7Eh	FSM_START7	FSM_START6	FSM_START5	FSM_START4	FSM_START3	FSM_START2	FSM_START1	FSM_START0
FSM_START_ADD_H	7Fh	FSM_START15	FSM_START714	FSM_START13	FSM_START12	FSM_START11	FSM_START10	FSM_START9	FSM_START8

2.3 Sensor hub registers

The table given below provides a list of the registers for the sensor hub functions available in the device and the corresponding addresses. The sensor hub registers are accessible when bit SHUB_REG_ACCESS is set to 1 in the FUNC_CFG_ACCESS register.

Table 5. Sensor hub registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SENSOR_HUB_1	02h	SensorHub1_7	SensorHub1_6	SensorHub1_5	SensorHub1_4	SensorHub1_3	SensorHub1_2	SensorHub1_1	SensorHub1_0
SENSOR_HUB_2	03h	SensorHub2_7	SensorHub2_6	SensorHub2_5	SensorHub2_4	SensorHub2_3	SensorHub2_2	SensorHub2_1	SensorHub2_0
SENSOR_HUB_3	04h	SensorHub3_7	SensorHub3_6	SensorHub3_5	SensorHub3_4	SensorHub3_3	SensorHub3_2	SensorHub3_1	SensorHub3_0
SENSOR_HUB_4	05h	SensorHub4_7	SensorHub4_6	SensorHub4_5	SensorHub4_4	SensorHub4_3	SensorHub4_2	SensorHub4_1	SensorHub4_0
SENSOR_HUB_5	06h	SensorHub5_7	SensorHub5_6	SensorHub5_5	SensorHub5_4	SensorHub5_3	SensorHub5_2	SensorHub5_1	SensorHub5_0
SENSOR_HUB_6	07h	SensorHub6_7	SensorHub6_6	SensorHub6_5	SensorHub6_4	SensorHub6_3	SensorHub6_2	SensorHub6_1	SensorHub6_0
SENSOR_HUB_7	08h	SensorHub7_7	SensorHub7_6	SensorHub7_5	SensorHub7_4	SensorHub7_3	SensorHub7_2	SensorHub7_1	SensorHub7_0
SENSOR_HUB_8	09h	SensorHub8_7	SensorHub8_6	SensorHub8_5	SensorHub8_4	SensorHub8_3	SensorHub8_2	SensorHub8_1	SensorHub8_0
SENSOR_HUB_9	0Ah	SensorHub9_7	SensorHub9_6	SensorHub9_5	SensorHub9_4	SensorHub9_3	SensorHub9_2	SensorHub9_1	SensorHub9_0
SENSOR_HUB_10	0Bh	SensorHub10_7	SensorHub10_6	SensorHub10_5	SensorHub10_4	SensorHub10_3	SensorHub10_2	SensorHub10_1	SensorHub10_0
SENSOR_HUB_11	0Ch	SensorHub11_7	SensorHub11_6	SensorHub11_5	SensorHub11_4	SensorHub11_3	SensorHub11_2	SensorHub11_1	SensorHub11_0
SENSOR_HUB_12	0Dh	SensorHub12_7	SensorHub12_6	SensorHub12_5	SensorHub12_4	SensorHub12_3	SensorHub12_2	SensorHub12_1	SensorHub12_0
SENSOR_HUB_13	0Eh	SensorHub13_7	SensorHub13_6	SensorHub13_5	SensorHub13_4	SensorHub13_3	SensorHub13_2	SensorHub13_1	SensorHub13_0
SENSOR_HUB_14	0Fh	SensorHub14_7	SensorHub14_6	SensorHub14_5	SensorHub14_4	SensorHub14_3	SensorHub14_2	SensorHub14_1	SensorHub14_0
SENSOR_HUB_15	10h	SensorHub15_7	SensorHub15_6	SensorHub15_5	SensorHub15_4	SensorHub15_3	SensorHub15_2	SensorHub15_1	SensorHub15_0
SENSOR_HUB_16	11h	SensorHub16_7	SensorHub16_6	SensorHub16_5	SensorHub16_4	SensorHub16_3	SensorHub16_2	SensorHub16_1	SensorHub16_0
SENSOR_HUB_17	12h	SensorHub17_7	SensorHub17_6	SensorHub17_5	SensorHub17_4	SensorHub17_3	SensorHub17_2	SensorHub17_1	SensorHub17_0
SENSOR_HUB_18	13h	SensorHub18_7	SensorHub18_6	SensorHub18_5	SensorHub18_4	SensorHub18_3	SensorHub18_2	SensorHub18_1	SensorHub18_0
MASTER_CONFIG	14h	RST_MASTER _REGS	WRITE_ONCE	START_CONFIG	PASS_ THROUGH_MODE	SHUB_PU_EN	MASTER_ON	AUX_SENS_ON1	AUX_SENS_ON0
SLV0_ADD	15h	slave0_add6	slave0_add5	slave0_add4	slave0_add3	slave0_add2	slave0_add1	slave0_add0	rw_0
SLV0_SUBADD	16h	slave0_reg7	slave0_reg6	slave0_reg5	slave0_reg4	slave0_reg3	slave0_reg2	slave0_reg1	slave0_reg0
SLAVE0_CONFIG	17h	SHUB_ODR1	SHUB_ODR0	0	0	BATCH_EXT _SENS_0_EN	Slave0_numop2	Slave0_numop1	Slave0_numop0
SLV1_ADD	18h	slave1_add6	slave1_add5	slave1_add4	slave1_add3	slave1_add2	slave1_add1	slave1_add0	r_1
SLV1_SUBADD	19h	slave1_reg7	slave1_reg6	slave1_reg5	slave1_reg4	slave1_reg3	slave1_reg2	slave1_reg1	slave1_reg0
SLAVE1_CONFIG	1Ah	0	0	0	0	BATCH_EXT _SENS_1_EN	Slave1_numop2	Slave1_numop1	Slave1_numop0
SLV2_ADD	1Bh	slave2_add6	slave1_add5	slave1_add4	slave1_add3	slave1_add2	slave1_add1	slave1_add0	r_2
SLV2_SUBADD	1Ch	slave2_reg7	slave2_reg6	slave2_reg5	slave2_reg4	slave2_reg3	slave2_reg2	slave2_reg1	slave2_reg0

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SLAVE2_CONFIG	1Dh	0	0	0	0	BATCH_EXT _SENS_2_EN	Slave2_numop2	Slave2_numop1	Slave2_numop0
SLV3_ADD	1Eh	slave3_add6	slave3_add5	slave3_add4	slave3_add3	slave3_add2	slave3_add1	slave3_add0	r_3
SLV3_SUBADD	1Fh	slave3_reg7	slave3_reg6	slave3_reg5	slave3_reg4	slave3_reg3	slave3_reg2	slave3_reg1	slave3_reg0
SLAVE3_CONFIG	20h	0	0	0	0	BATCH_EXT _SENS_3_EN	Slave3_numop2	Slave3_numop1	Slave3_numop0
DATAWRITE_SLV0	21h	Slave0_dataw7	Slave0_dataw6	Slave0_dataw5	Slave0_dataw4	Slave0_dataw3	Slave0_dataw2	Slave0_dataw1	Slave0_dataw0
STATUS_MASTER	22h	WR_ONCE_DONE	SLAVE3_NACK	SLAVE2_NACK	SLAVE1_NACK	SLAVE0_NACK	0	0	SENS_HUB _ENDOP

3 Operating modes

The IIS2ICLX provides two possible operating configurations:

- Power-down mode;
- Normal mode.

The device offers a wide VDD voltage range from 1.71 V to 3.6 V and a VDDIO range from 1.62 V to 3.6 V. The power-on sequence is not restricted: VDD/VDDIO pins can be either set to power supply level or to ground level (they must not be left floating) and no specific sequence is required for powering them on.

In order to avoid potential conflicts, during the power-on sequence it is recommended to set the lines (on the host side) connected to the device IO pins floating or connected to ground until VDDIO is set. After VDDIO is set, the IO pins have to be configured according to their default status described in Table 1. Pin status. In order to avoid an unexpected increase in current consumption the input pins which are not pulled-up/pulled-down must the polarized by the host.

When the VDD power supply is applied, the device performs a 10 ms (maximum) boot procedure to load the trimming parameters. After the boot is completed, the accelerometer is automatically configured in Power-Down mode. To guarantee proper power-off of the device it is recommended to maintain the duration of the VDD line to GND for at least 100 µs.

When the sensor is in Power-Down mode, almost all internal blocks of the device are switched off. The SPI / I²C digital interface remains active to allow communication with the device. The content of the configuration registers is preserved and the output data registers are not updated, keeping the last data sampled in memory before going into Power-Down mode.

Referring to the datasheet, the output data rate (ODR_XL) bits of the CTRL1_XL register are used to select the power mode and the output data rate of the accelerometer (Table 6. Accelerometer ODR and power mode selection).

Table 6. Accelerometer ODR and power mode selection

ODR_XL [3:0]	ODR [Hz]
0000	Power-Down
0001	12.5 Hz
0010	26 Hz
0011	52 Hz
0100	104 Hz
0101	208 Hz
0110	416 Hz
0111	833 Hz

When in Normal mode, the device current consumption is 420 µA (typ. @ Vdd = 3.0 V, T = 25 °C) regardless of the selected output data rate.

3.1 Connection modes

The device offers two different connection modes, described in detail in this document:

- **Mode 1:** it is the connection mode enabled by default; I²C slave interface or SPI (3- / 4-wire) serial interface is available.
- **Mode 2:** it is the sensor hub mode; I²C slave interface or SPI (3- / 4-wire) serial interface and I²C interface master for external sensor connections are available. This connection mode is described in Section 7 Mode 2 - Sensor hub mode.

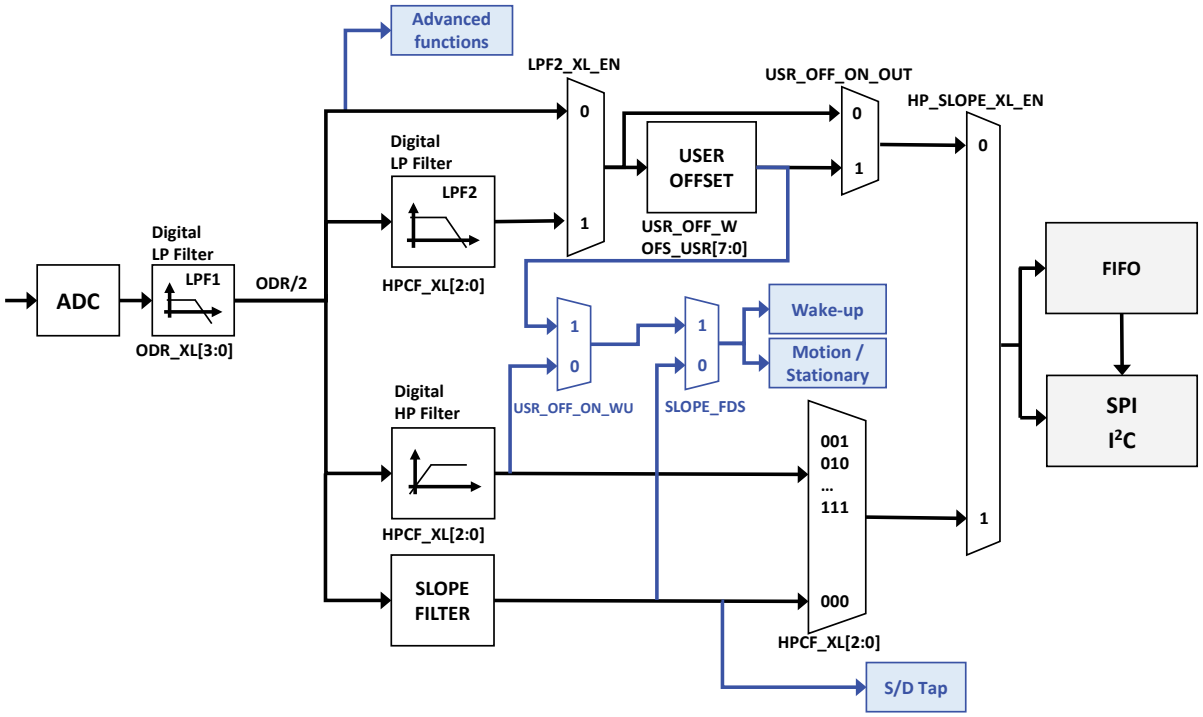
3.2 Accelerometer filtering chain

The accelerometer sampling chain is represented by a cascade of three main blocks: an ADC converter, a digital low-pass filter (LPF1) and the composite group of digital filters.

Figure 2 shows the accelerometer sampling chain.

The analog signal coming from the mechanical parts is converted by the ADC; then, the digital LPF1 filter provides different cutoff values according to the selected sampling rate.

Figure 2. Accelerometer filtering chain



The “Advanced functions” block in the figure above refers to the Finite State Machine and the Machine Learning Core.

Finally, the composite group of filters composed of a low-pass digital filter (LPF2), a high-pass digital filter and a slope filter processes the digital signal.

The LPF2_XL_EN bit of CTRL1_XL register and the CTRL8_XL register can be used to configure the composite filter group and the overall bandwidth of the accelerometer filtering chain, as shown in Table 7. Accelerometer bandwidth selection. Referring to this table, on the low-pass path side, the Bandwidth columns refer to the LPF1 bandwidth if LPF2_XL_EN = 0; they refer to the LPF2 bandwidth if LPF2_XL_EN = 1. On the high-pass path side, the Bandwidth columns refer to the Slope filter bandwidth if HPCF_XL[2:0] = 000b; they refer to the HP filter bandwidth for all the other configurations.

Table 7. Accelerometer bandwidth selection also provides the maximum (worst case) settling time in terms of samples to be discarded for the various configurations of the accelerometer filtering chain.

Table 7. Accelerometer bandwidth selection

HP_SLOPE_XL_EN	LPF2_XL_EN	HPCF_XL[2:0]	Bandwidth	Max overall settling time[1] (samples to be discarded)
0 (Low-pass path)	0	-	ODR / 2	4
	1	000	ODR / 4	4
		001	ODR / 10	10
		010	ODR / 20	19
		011	ODR / 45	38
		100	ODR / 100	75
		101	ODR / 200	150
		110	ODR / 400	296
		111	ODR / 800	595
1 (High-pass path)	-	000	ODR / 4 (slope filter)	4
		001	ODR / 10	14
		010	ODR / 20	19
		011	ODR / 45	38
		100	ODR / 100	75
		101	ODR / 200	150
		110	ODR / 400	296
		111	ODR / 800	595

1. *Settling time @ 99% of the final value, taking into account all output data rates and all operating mode switches*

Setting the HP_SLOPE_XL_EN bit to 0, the low-pass path of the composite filter block is selected. If the LPF2_XL_EN bit is set to 0, no additional filter is applied; if the LPF2_XL_EN bit is set to 1, the LPF2 filter is applied in addition to LPF1 and the overall bandwidth of the accelerometer chain can be set by configuring the HPCF_XL[2:0] field of the CTRL8_XL register.

Setting the HP_SLOPE_XL_EN bit to 1, the high-pass path of the composite filter block is selected: the HPCF_XL[2:0] field is used in order to enable, in addition to the LPF1 filter, either the Slope filter usage (when HPCF_XL[2:0] = 000b) or the digital High-Pass filter (other HPCF_XL[2:0] configurations). The HPCF_XL[2:0] field is also used to select the cutoff frequencies of the HP filter.

The high-pass filter reference mode feature is available for the accelerometer sensor: when this feature is enabled, the current X and Y accelerometer sample is internally stored and subtracted from all subsequent output values. In order to enable the reference mode, both the HP_REF_MODE_XL bit and the HP_SLOPE_XL_EN bit of the CTRL8_XL register have to be set to 1, and the value of the HPCF_XL[2:0] field must be equal to 111b. When the reference mode feature is enabled, both the LPF2 filter and the HP filter are not available. The first accelerometer output data after enabling the reference mode has to be discarded.

The FASTSETTL_MODE_XL bit of CTRL8_XL register enables the accelerometer LPF2 or HPF fast-settling mode: the selected filter sets the second sample after writing this bit. This feature applies only upon device exit from Power-Down mode.

3.2.1 Accelerometer slope filter

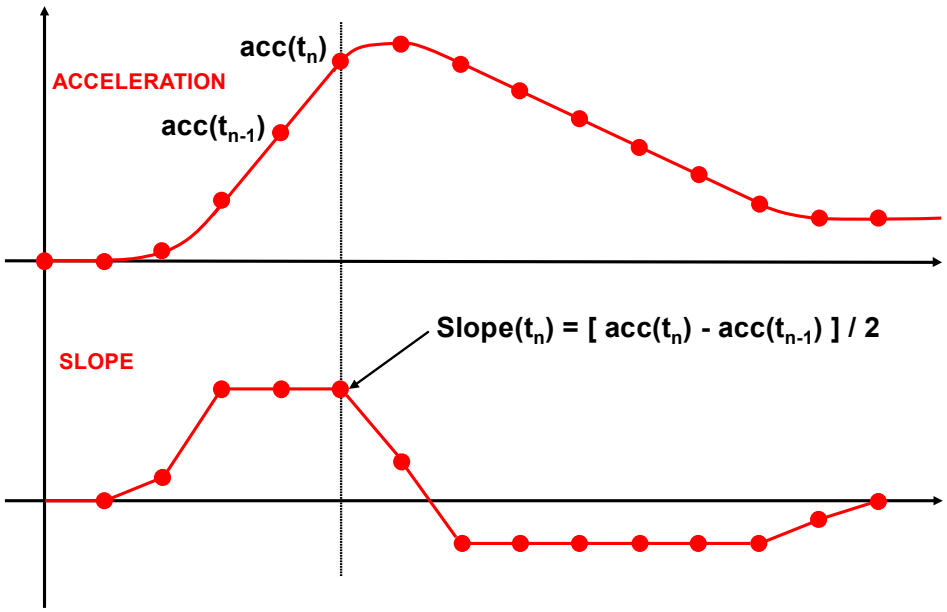
As shown in Figure 3. Accelerometer slope filter, the device embeds a digital slope filter, which can also be used for some embedded features such as single/double-tap recognition, wake-up detection and motion/stationary.

The slope filter output data is computed using the following formula:

$$slope(t_n) = [\ acc(t_n) - acc(t_{n-1})\ ] / 2$$

An example of a slope data signal is illustrated in the following figure.

Figure 3. Accelerometer slope filter



3.3 Accelerometer turn-on/off time

The accelerometer reading chain contains low-pass filtering to improve signal-to-noise performance. For this reason, it is necessary to take into account the settling time of the filters when switching the accelerometer operating mode.

Table 7. Accelerometer bandwidth selection provides the maximum (worst case) turn-on time (when switching from Power-Down mode to Normal mode) in terms of samples to be discarded for the various configurations ofthe accelerometer filtering chain.

Maximum turn-off time when switching from Normal mode to Power-Down mode is 1 µs.

4 Mode 1 - Reading output data

4.1 Startup sequence

Once the device is powered up, it automatically downloads the calibration coefficients from the embedded flash to the internal registers. When the boot procedure is completed, i.e. after approximately 10 milliseconds, the accelerometer automatically enters Power-Down mode.

To turn on the accelerometer and gather acceleration data through the I²C / SPI interface, it is necessary to select one of the output data rates through the CTRL1_XL register.

The following general-purpose sequence can be used to configure the accelerorometer:

1. Write INT1_CTRL = 01h // Acc data-ready interrupt on INT1
2. Write CTRL1_XL = 60h // Acc = 416 Hz (Normal mode)

4.2 Using the status register

The device is provided with a STATUS_REG register which should be polled to check when a new set of data is available. The XLDA bit is set to 1 when a new set of data is available at the accelerometer output.

For the accelerometer, the read of the output registers should be performed as follows:

1. Read STATUS_REG
2. If XLDA = 0, then go to 1
3. Read OUTX_L_A
4. Read OUTX_H_A
5. Read OUTY_L_A
6. Read OUTY_H_A
7. Data processing
8. Go to 1

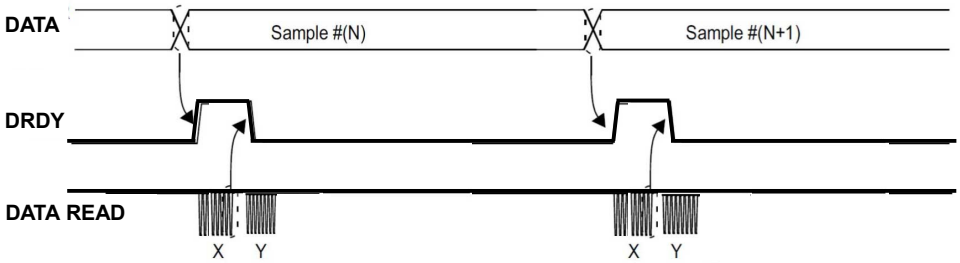
4.3 Using the data-ready signal

The device can be configured to have one hardware signal to determine when a new set of measurement data is available to be read.

The accelerometer sensor data-ready signal is represented by the XLDA bit of the STATUS_REG register. The signal can be driven to the INT1 pin by setting the INT1_DRDY_XL bit of the INT1_CTRL register to 1 and to the INT2 pin by setting the INT2_DRDY_XL bit of the INT2_CTRL register to 1.

The data-ready signal rises to 1 when a new set of data has been generated and it is available to be read. The data-ready signal can be either latched or pulsed: if the dataready_pulsed bit of the COUNTER_BDR_REG1 register is set to 0 (default value), then the data-ready signal is latched and the interrupt is reset when the higher part of one of the enabled channels is read (registers 29h, 2Bh). If the dataready_pulsed bit of the COUNTER_BDR_REG1 register is set to 1, then the data-ready is pulsed and the duration of the pulse observed on the interrupt pins is 75 µs. Pulsed mode is not applied to the XLDA bit which is always latched.

Figure 4. Data-ready signal



4.3.1 DRDY mask functionality

Setting the DRDY_MASK bit of the CTRL4_C register to 1, the accelerometer data-ready signal is masked until the settling of the sensor filters is completed.

When FIFO is active and the DRDY_MASK bit is set to 1, accelerometer invalid samples stored in FIFO are equal to 7FFDh. In this way, a tag is applied to the invalid samples stored in the FIFO buffer so that they can be easily identified and discarded during data post-processing.

Note: The DRDY_MASK bit acts only on the accelerometer LPF1 digital filter settling time.

4.4 Using the block data update (BDU) feature

If reading the accelerometer data is particularly slow and cannot be synchronized (or it is not required) with the XLDA bit in the STATUS_REG register or with the DRDY signal driven to the INT1/INT2 pins, it is strongly recommended to set the BDU (Block Data Update) bit to 1 in the CTRL3_C register.

This feature avoids reading values (most significant and least significant parts of output data) related to different samples. In particular, when the BDU is activated, the data registers related to each channel always contain the most recent output data produced by the device, but, in case the read of a given pair (i.e. OUTX_H_A and OUTX_L_A, OUTY_H_A and OUTY_L_A) is initiated, the refresh for that pair is blocked until both MSB and LSB parts of the data are read.

Note: BDU only guarantees that the LSB part and MSB part have been sampled at the same moment. For example, if the reading speed is too slow, X can be read at T1 and Y sampled at T2.

The BDU feature also acts on the FIFO_STATUS1 and FIFO_STATUS2 registers. When the BDU bit is set to 1, it is mandatory to read FIFO_STATUS1 first and then FIFO_STATUS2.

4.5 Understanding output data

The measured acceleration data are sent to the OUTX_H_A, OUTX_L_A and OUTY_H_A, OUTY_L_A registers. These registers contain, respectively, the most significant part and the least significant part of the acceleration signals acting on the X and Y axes.

The complete output data for the X and Y channels is given by the concatenation of OUTX_H_A & OUTX_L_A, OUTY_H_A & OUTY_L_A and it is expressed as a two's complement number.

The acceleration data are represented as 16-bit numbers.

4.5.1 Examples of output data

Table 8. Content of output data registers vs. acceleration (FS_XL = ±2 *g*) provides a few basic examples of the accelerometer data that is read in the data registers when the device is subjected to a given acceleration.

The values listed in the following tables are given under the hypothesis of perfect device calibration (i.e. no offset, no gain error, etc.).

Table 8. Content of output data registers vs. acceleration (FS_XL = ±2 *g*)

Acceleration values	Register address	
	OUTX_H_A (29h)	OUTX_L_A (28h)
0 *g*	00h	00h
350 m*g*	16h	69h
1 *g*	40h	09h
-350 m*g*	E9h	97h
-1 *g*	BFh	F7h

4.6 Accelerometer offset registers

The device provides accelerometer offset registers (X_OFS_USR, Y_OFS_USR) which can be used for zero-*g* offset correction or, in general, to apply an offset to the accelerometer output data.

The accelerometer offset block can be enabled by setting the USR_OFF_ON_OUT bit of the CTRL7_XL register. The offset value set in the offset registers is internally subtracted from the measured acceleration value for the respective axis; internally processed data are then sent to the accelerometer output register and to the FIFO (if enabled). These register values are expressed as an 8-bit word in two's complement and must be in the range [-127, 127].

The weight [g/LSB] to be applied to the offset register values is independent of the accelerometer selected full scale and can be configured using the USR_OFF_W bit of the CTRL6_C register:

- 2^{-10}*g*/LSB if the USR_OFF_W bit is set to 0;
- 2^{-6} *g*/LSB if the USR_OFF_W bit is set to 1.

4.7 Rounding functions

The rounding function can be used to auto address the device registers for a circular burst-mode read. Basically, with a multiple read operation the address of the register that is being read goes automatically from the first register to the last register of the pattern and then goes back to the first one.

The rounding function is automatically enabled when performing a multiple read operation of the FIFO output registers: after reading FIFO_DATA_OUT_Z_H (7Eh), the address of the next register that will be read goes automatically back to FIFO_DATA_OUT_TAG (78h), allowing the user to read many data with a unique multiple read.

4.8 DEN (data enable)

The device allows an external trigger level recognition by enabling the TRIG_EN, LVL1_EN, LVL2_EN bits in CTRL6_C register.

Four different modes can be selected (see Table 9. DEN configurations):

- Edge-sensitive trigger mode;
- Level-sensitive trigger mode;
- Level-sensitive latched mode;
- Level-sensitive FIFO enable mode.

The Data Enable (DEN) input signal must be driven on the INT2 pin, which is configured as an input pin when one of these modes is enabled.

The DEN functionality is disabled by default. In order to enable it, the bit DEN_EN in the CTRL4_C register must be set to 1.

The DEN active level is low by default. It can be changed to active-high by setting the bit DEN_LH in CTRL5_C register to 1.

Table 9. DEN configurations

TRIG_EN	LVL1_EN	LVL2_EN	Function	Trigger type	Action
0	0	0	Data enable off	-	-
1	0	0	Edge-sensitive trigger mode	Edge	Data generation
0	1	0	Level-sensitive trigger mode	Level	Data stamping
0	1	1	Level-sensitive latched mode	Edge	Data stamping
1	1	0	Level-sensitive FIFO enable mode	Level	Data generation in FIFO and stamping

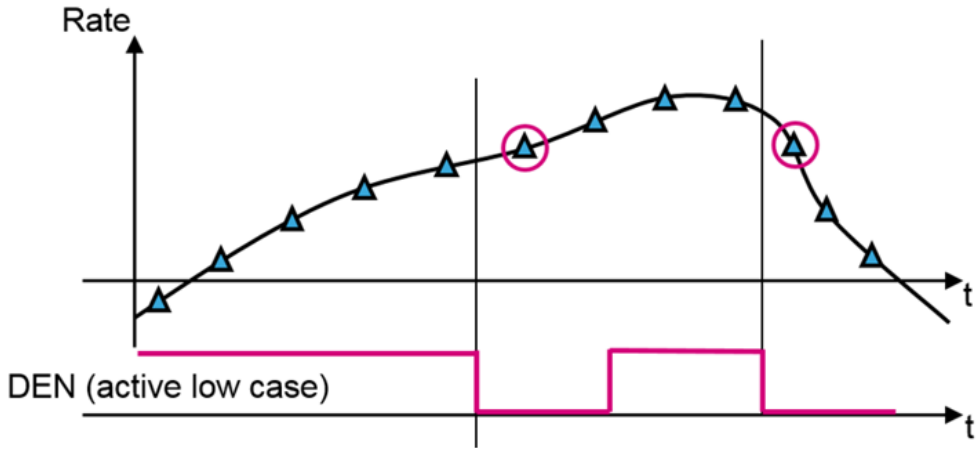
4.8.1 Edge-sensitive trigger mode

Edge-sensitive trigger mode can be enabled by setting the TRIG_EN bit in CTRL6_C to 1, and LVL1_EN, LVL2_EN bits in CTRL6_C register to 0.

Once the edge-sensitive trigger mode is enabled, the FIFO buffer and output registers are filled with the first sample acquired after every rising edge (if DEN_LH bit is equal to 1) or falling edge (if DEN_LH bit is equal to 0) of the DEN input signal.

The following figure shows, with red circles, the samples acquired after the falling edges (DEN active-low).

Figure 5. Edge-sensitive trigger mode, DEN active-low



Please note that the DEN level is internally read just before the update of the data registers: if a level change occurs after the read, DEN will be acknowledged in the next ODR.

If the edge-sensitive trigger mode is enabled and the accelerometer is stored in FIFO, the following limitations must be implemented:

- Accelerometer batch data rate (BDR_XL_[3:0] bits of the FIFO_CTRL3 register) and accelerometer output data rate (ODR_XL[3:0] of the CTRL1_XL register) must be set to the same value;
- Configuration-change sensor (CFG-Change) is not allowed (ODRCHG_EN bit of the FIFO_CTRL2 register must be set to 0);
- Timestamp decimation in FIFO is not allowed (DEC_TS_BATCH_[1:0] bits of the FIFO_CTRL4 register must be set to 00b).

In the example shown below, the FIFO has been configured to store the accelerometer data only; when the DEN signal toggles, the data are written to FIFO on the falling edge.

1.	Write 04h to FIFO_CTRL3	// Enable accelerometer in FIFO @ 104 Hz
2.	Write 06h to FIFO_CTRL4	// Set FIFO in Continuous mode
		// Enable the edge-sensitive trigger
3.	Write 80h to CTRL6_C	// INT2 pin is switched to input mode (DEN signal)
4.	Write FAh to CTRL9_XL	// Enable DEN functionality
		// DEVICE_CONF = 1
		// Select DEN active level (active low)
5.	Write 40h to CTRL1_XL	// Turn on the accelerometer: ODR_XL = 104 Hz, FS_XL = ±0.5 *g*

4.8.2 Level-sensitive trigger mode

Level-sensitive trigger mode can be enabled by setting the LVL1_EN bit in the CTRL6_C register to 1, and the TRIG_EN, LVL2_EN bits in the CTRL6_C register to 0.

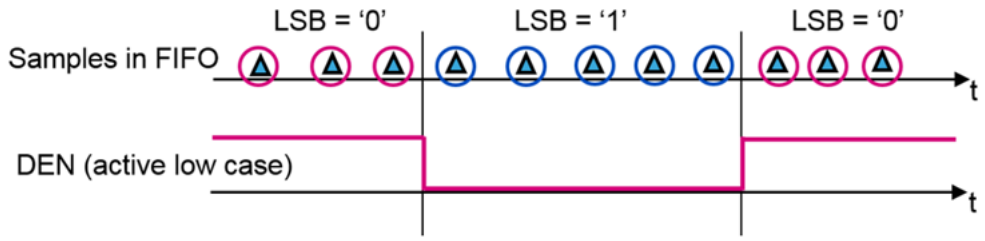
Once the level-sensitive trigger mode is enabled, the LSB bit of the selected data (in output registers and FIFO) is replaced by 1 if the DEN level is active, or 0 if the DEN level is not active. The selected data can be the X and/or Y axis of the accelerometer sensor (see Section 4.8.5 LSB selection for DEN stamping for details).

Accelerometer data can be stored in the FIFO according to the FIFO settings.

Please note that the DEN level is internally read just before the update of the data registers: if a level change occurs after the read, DEN will be acknowledged in the next ODR.

Figure 6 shows with red circles the samples stored in the FIFO with LSB = 0 (DEN not active) and with blue circles the samples stored in the FIFO with LSB = 1 (DEN active).

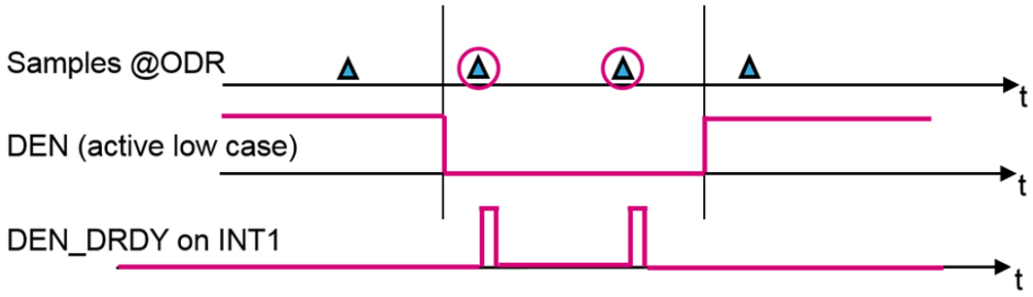
Figure 6. Level-sensitive trigger mode, DEN active-low



When the level-sensitive trigger mode is enabled, the DEN signal can also be used to filter the data-ready signal on the INT1 pin. INT1 will show data-ready information only when the DEN pin is in the active state. To do this, the bit DEN_DRDY_flag of the INT1_CTRL register must be set to 1. The interrupt signal can be latched or pulsed according to the dataready_pulsed bit of the COUNTER_BDR_REG1 register.

Figure 7 shows an example of data-ready on INT1 when the DEN level is low (active state).

Figure 7. Level-sensitive trigger mode, DEN active-low, DEN_DRDY on INT1



4.8.3 Level-sensitive latched mode

Level-sensitive latched mode can be enabled by setting the LVL1_EN and LVL2_EN bits in the CTRL6_C register to 1, and the TRIG_EN bit in the CTRL6_C register to 0.

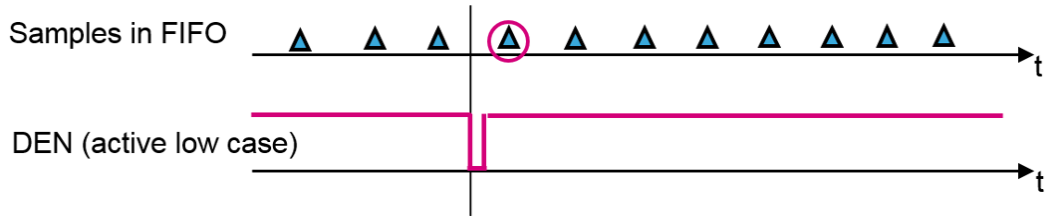
When the level-sensitive latched mode is enabled, the LSB bit of the selected data (in output registers and FIFO) is normally set to 0 and becomes 1 only on the first sample after a pulse on the DEN pin.

Please note that the DEN level is internally read just before the update of the data registers: if a level change occurs after the read, DEN will be acknowledged in the next ODR.

Data can be selected through the DEN_X and DEN_Y bits in CTRL9_XL (see Section 4.8.5 LSB selection for DEN stamping for details).

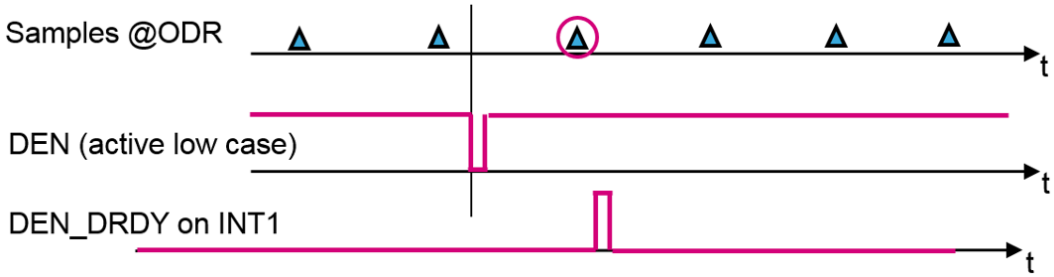
Figure 8 shows an example of level-sensitive latched mode with DEN active-low. After the pulse on the DEN pin, the sample with a red circle will have the value 1 on the LSB bit. All the other samples will have LSB bit 0.

Figure 8. Level-sensitive latched mode, DEN active-low



When the level-sensitive latched mode is enabled and the bit DEN_DRDY_flag of the INT1_CTRL register is set to 1, a pulse is generated on the INT1 pin corresponding to the availability of the first sample generated after the DEN pulse occurrence (see Figure 9).

Figure 9. Level-sensitive latched mode, DEN active-low, DEN_DRDY on INT1



4.8.4 Level-sensitive FIFO enable mode

Level-sensitive FIFO enable mode can be enabled by setting the TRIG_EN and LVL1_EN bits in the CTRL6_C register to 1, and the LVL2_EN bit in the CTRL6_C register to 0.

Once the level-sensitive FIFO enable mode is enabled, data is stored in the FIFO only when the DEN pin is equal to the active state.

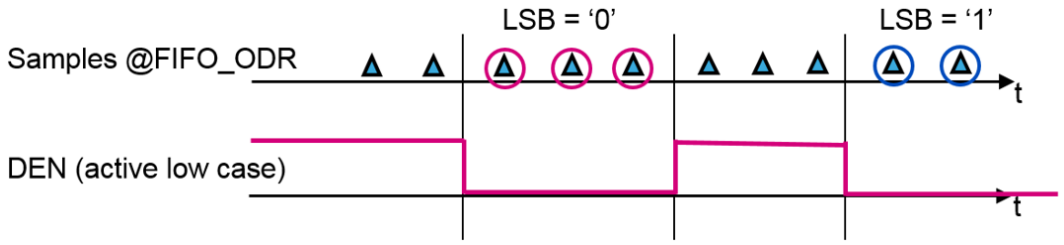
In this mode, the LSB bit of the selected data (in output registers and FIFO) is replaced by 0 for odd DEN events and by 1 for even DEN events. This feature allows distinguishing the data stored in FIFO during the current DEN active window from the data stored in FIFO during the next DEN active window.

Please note that the DEN level is internally read just before the update of the data registers: if a level change occurs after the reading, DEN will be acknowledged in the next ODR.

The selected data can be the X and/or Y axis of the accelerometer sensor. Data can be selected through the DEN_X and DEN_Y bits in the CTRL9_XL register (see Section 4.8.5 LSB selection for DEN stamping for details).

An example of level-sensitive FIFO enable mode is shown in Figure 10, the red circles show the samples stored in the FIFO with LSB bit 0, while the blue circles show the samples with LSB bit 1.

Figure 10. Level-sensitive FIFO enable mode, DEN active-low



When using level-sensitive FIFO enabled mode, some limitations must be taken into account in the FIFO configuration:

- Accelerometer batch data rate (BDR_XL_[3:0] bits of the FIFO_CTRL3 register) and accelerometer output data rate (ODR_XL[3:0] of the CTRL1_XL register) must be set to the same value;
- Configuration-change sensor (CFG-Change) is not allowed (ODRCHG_EN bit of the FIFO_CTRL2 register must be set to 0);
- Timestamp decimation in FIFO is not allowed (DEC_TS_BATCH_[1:0] bits of the FIFO_CTRL4 register must be set to 00b).

4.8.5 LSB selection for DEN stamping

When level-sensitive modes (trigger or latched) are used, it is possible to select which LSB have to contain the information related to DEN pin behavior. This information can be stamped on the accelerometer axis in accordance with bits DEN_X and DEN_Y of the CTRL9_XL register. Setting to 1 the DEN_X and DEN_Y bits, DEN information is stamped in the LSB of the corresponding axes.

By default, the bits are configured to have information on both the accelerometer axes.

5 Interrupt generation

Interrupt generation is based on accelerometer data, so, for interrupt-generation purposes, the accelerometer sensor has to be set in Normal mode.

The interrupt generator can be configured to detect:

- Wake-up;
- Single-tap and double-tap sensing;
- Motion/Stationary recognition.

Moreover, the device can be configured to generate interrupt signals activated by user-defined motion patterns. To do this, up to 16 embedded finite state machines can be programmed independently for motion detection. Furthermore up to 8 decision trees can simultaneously and independently run inside the Machine Learning Core logic.

The embedded Finite State Machine and the Machine Learning Core features offer very high customization capabilities starting from scratch or importing configurations directly provided by STMicroelectronics. Please refer to the Finite State Machine application note and the Machine Learning Core application note available on www.st.com.

All these interrupt signals, together with the FIFO interrupt signals, can be independently driven to the INT1 and INT2 interrupt pins or checked by reading the dedicated source register bits.

The H_LACTIVE bit of the CTRL3_C register must be used to select the polarity of the interrupt pins. If this bit is set to 0 (default value), the interrupt pins are active high and they change from low to high level when the related interrupt condition is verified. Otherwise, if the H_LACTIVE bit is set to 1 (active low), the interrupt pins are normally at high level and they change from high to low when interrupt condition is reached.

The PP_OD bit of CTR3_C allows changing the behavior of the interrupt pins from push-pull to open drain. If the PP_OD bit is set to 0, the interrupt pins are in push-pull configuration (low-impedance output for both high and low level). When the PP_OD bit is set to 1, only the interrupt active state is a low-impedance output.

5.1 Interrupt pin configuration

The device is provided with two pins that can be activated to generate either data-ready or interrupt signals. The functionality of these pins is selected through the MD1_CFG and INT1_CTRL registers for the INT1 pin, and through the MD2_CFG and INT2_CTRL registers for the INT2 pin.

A brief description of these interrupt control registers is given in the following summary; the default value of their bits is equal to 0, which corresponds to ‘disable’. In order to enable the routing of a specific interrupt signal on the pin, the related bit has to be set to 1.

Table 10. INT1_CTRL register

b7	b6	b5	b4	b3	b2	b1	b0
DEN_DRDY _flag	INT1_ CNT_BDR	INT1_ FIFO_FULL	INT1_ FIFO_OVR	INT1_ FIFO_TH	INT1_ BOOT	0	INT1_ DRDY_XL

- DEN_DRDY_flag: DEN_DRDY flag interrupt on INT1
- INT1_CNT_BDR: FIFO COUNTER_BDR_IA interrupt on INT1
- INT1_FIFO_FULL: FIFO full flag interrupt on INT1
- INT1_FIFO_OVR: FIFO overrun flag interrupt on INT1
- INT1_FIFO_TH: FIFO threshold interrupt on INT1
- INT1_BOOT: Boot interrupt on INT1
- INT1_DRDY_XL: Accelerometer data-ready on INT1

Table 11. MD1_CFG register

b7	b6	b5	b4	b3	b2	b1	b0
INT1_SLEEP _CHANGE	INT1_ SINGLE_TAP	INT1_WU	0	INT1_ DOUBLE_TAP	0	INT1_ EMB_FUNC	INT1_ SHUB

- INT1_SLEEP_CHANGE: Motion/stationary recognition event interrupt on INT1
- INT1_SINGLE_TAP: Single-tap interrupt on INT1
- INT1_WU: Wake-up interrupt on INT1
- INT1_DOUBLE_TAP: Double-tap interrupt on INT1
- INT1_EMB_FUNC: embedded functions interrupt on INT1
- INT1_SHUB: sensor hub end operation interrupt on INT1

Table 12. INT2_CTRL register

b7	b6	b5	b4	b3	b2	b1	b0
0	INT2_ CNT_BDR	INT2_ FIFO_FULL	INT2_ FIFO_OVR	INT2_ FIFO_TH	INT2_ DRDY_TEMP	0	INT2_ DRDY_XL

- INT2_CNT_BDR: FIFO COUNTER_BDR_IA interrupt on INT2
- INT2_FIFO_FULL: FIFO full flag interrupt on INT2
- INT2_FIFO_OVR: FIFO overrun flag interrupt on INT2
- INT2_FIFO_TH: FIFO threshold interrupt on INT2
- INT2_DRDY_TEMP: Temperature data-ready on INT2
- INT2_DRDY_XL: Accelerometer data-ready on INT2

Table 13. MD2_CFG register

b7	b6	b5	b4	b3	b2	b1	b0
INT2_SLEEP _CHANGE	INT2_ SINGLE_TAP	INT2_WU	0	INT2_ DOUBLE_TAP	0	INT2_ EMB_FUNC	INT2_ TIMESTAMP

- INT2_SLEEP_CHANGE: Motion/stationary recognition event interrupt on INT2
- INT2_SINGLE_TAP: Single-tap interrupt on INT2
- INT2_WU: Wake-up interrupt on INT2
- INT2_DOUBLE_TAP: Double-tap interrupt on INT2
- INT2_EMB_FUNC: embedded functions interrupt on INT2
- INT2_TIMESTAMP: timestamp overflow alert interrupt on INT2

If multiple interrupt signals are routed on the same pin (INTx), the logic level of this pin is the “OR” combination of the selected interrupt signals. In order to know which event has generated the interrupt condition, the related source registers have to be read:

- WAKE_UP_SRC, TAP_SRC (basic interrupt functions)
- STATUS_REG (for data-ready signals)
- EMBD_FUNC_STATUS_MAINPAGE / EMB_FUNC_SRC (for embedded functions)
- FSM_STATUS_A_MAINPAGE / FSM_STATUS_A and FSM_STATUS_B_MAINPAGE / FSM_STATUS_B (for Finite State Machine)
- STATUS_MASTER_MAINPAGE / STATUS_MASTER (for sensor-hub)
- FIFO_STATUS2 (for FIFO).

The ALL_INT_SRC register groups the basic interrupts functions event status (wake-up, tap, motion/stationary) in a single register: it is possible to read this register in order to address a subsequent specific source register read.

The INT2_on_INT1 pin of CTRL4_C register allows driving all the enabled interrupt signals in logic “OR” on the INT1 pin (by setting this bit to 1). When this bit is set to 0, the interrupt signals are divided between the INT1 and INT2 pins.

The basic interrupts have to be enabled by setting the INTERRUPTS_ENABLE bit in the TAP_CFG2 register.

The LIR bit of the TAP_CFG0 register enables the latched interrupt for the basic interrupt functions: when this bit is set to 1 and the interrupt flag is sent to the INT1 pin and/or INT2 pin, the interrupt remains active until the ALL_INT_SRC register or the corresponding source register is read, and it is reset at the next ODR cycle. The latched interrupt is enabled on a function only if a function is routed to the INT1 or INT2 pin: if latched mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

Note: If latched mode is enabled (LIR = 1), it is not recommended to continuously poll the ALL_INT_SRC or the dedicated source registers, because by reading them the embedded functions are internally reset; a synchronous (with interrupt event) read of the source registers is recommended in this case.

When latched mode is enabled (LIR=1), it is possible to force the immediate reset of the interrupt signal routed on the INT1 or INT2 pin and its corresponding interrupt status bit when ALL_INT_SRC (or the related source register) is read. In order to perform this immediate reset, the INT_CLR_ON_READ bit of the TAP_CFG0 register must be set to 1. When bit INT_CLR_ON_READ is equal to 0, the reset occurs at the next ODR cycle.

5.2 Wake-up interrupt

The wake-up feature can be implemented using either the slope filter (see Section 3.2.1 Accelerometer slope filter for more details) or the high-pass digital filter, as illustrated in Figure 2. Accelerometer filtering chain. The filter to be applied can be selected using the SLOPE_FDS bit of the TAP_CFG0 register: if this bit is set to 0 (default value), the slope filter is used; if it’s set to 1, the HPF digital filter is used. Moreover, it is possible to configure the wake-up feature as an absolute wake-up with respect to a programmable position. This can be done by setting both the SLOPE_FDS bit of the TAP_CFG0 register and the USR_OFF_ON_WU bit of the WAKE_UP_THS register to 1. Using this configuration, the input data for the wake-up function comes from the low-pass filter path and the programmable position is subtracted as an offset. The programmable position can be configured through the X_OFS_USR and Y_OFS_USR and registers (refer to Section 4.6 Accelerometer offset registers for more details).

The wake-up interrupt signal is generated if a certain number of consecutive filtered data exceed the configured threshold (Figure 11. Wake-up interrupt (using the slope filter)).

The unsigned threshold value is defined using the WK_THS[5:0] bits of the WAKE_UP_THS register; the value of 1 LSB of these 6 bits depends on the selected accelerometer full scale and on the value of the WAKE_THS_W bit of the WAKE_UP_DUR register:

- If WAKE_THS_W = 0, 1 LSB = FS_XL / 2^6;
- If WAKE_THS_W = 1, 1 LSB = FS_XL / 2^8.

Note: If selecting FS_XL = ±3 g:

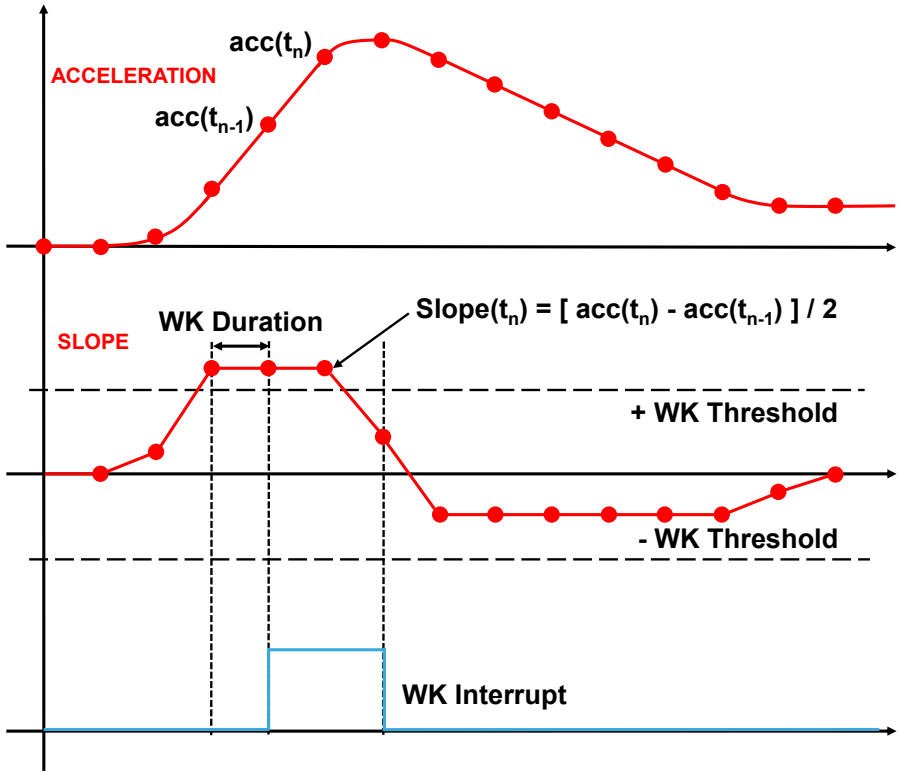
- *If WAKE_THS_W = 0, 1 LSB = 4 / 2^6 g;*
- *If WAKE_THS_W = 1, 1 LSB = 4 / 2^8 g.*

The threshold is applied to both positive and negative data: for wake-up interrupt generation, the absolute value of the filtered data must be bigger than the threshold.

The duration parameter defines the minimum duration of the wake-up event to be recognized; its value is set using the WAKE_DUR[1:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to 1/ODR_XL time, where ODR_XL is the accelerometer output data rate. It is important to appropriately define the duration parameter to avoid unwanted wake-up interrupts due to spurious spikes of the input signal.

This interrupt signal can be enabled by setting the INTERRUPTS_ENABLE bit in the TAP_CFG2 register to 1 and can be driven to the two interrupt pins by setting to 1 the INT1_WU bit of the MD1_CFG register or the INT2_WU bit of the MD2_CFG register; it can also be checked by reading the WU_IA bit of the WAKE_UP_SRC or ALL_INT_SRC register. The X_WU and Y_WU bits of the WAKE_UP_SRC register indicate which axes have triggered the wake-up event.

Figure 11. **Wake-up interrupt (using the slope filter)**



If latch mode is disabled (LIR bit of TAP_CFG0 is set to 0), the interrupt signal is automatically reset when the filtered data falls below the threshold. If latch mode is enabled and the wake-up interrupt signal is driven to the interrupt pins, once a wake-up event has occurred and the interrupt pin is asserted, it must be reset by reading the WAKE_UP_SRC register or the ALL_INT_SRC register. The X_WU and Y_WU bits are maintained at the state in which the interrupt was generated until the read is performed, and released at the next ODR cycle. In case the WU_X and WU_Y bits have to be evaluated (in addition to the WU_IA bit), it is recommended to directly read the WAKE_UP_SRC register (do not use ALL_INT_SRC register for this specific case). If latch mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

A basic SW routine for wake-up event recognition using the high-pass digital filter is given below.

1. Write 6Ch to CTRL1_XL // Turn on the accelerometer
   // ODR_XL = 416 Hz, FS_XL = ±2 *g*
2. Write 51h to TAP_CFG0 // Enable latch mode with reset on read and digital high-pass filter
3. Write 80h to TAP_CFG2 // Enable interrupt function
4. Write 00h to WAKE_UP_DUR // No duration and selection of wake-up threshold weight (1 LSB = FS_XL / 2^6)
5. Write 02h to WAKE_UP_THS // Set wake-up threshold
6. Write 20h to MD1_CFG // Wake-up interrupt driven to INT1 pin

Since the duration time is set to zero, the wake-up interrupt signal is generated for each X, Y filtered data exceeding the configured threshold. The WK_THS field of the WAKE_UP_THS register is set to 000010b, therefore the wake-up threshold is 62.5 mg (= 2 * FS_XL / 2^6).

Since the wake-up functionality is implemented using the slope/high-pass digital filter, it is necessary to consider the settling time of the filter just after this functionality is enabled. For example, when using the slope filter (but a similar consideration can be done for the high-pass digital filter usage) the wake-up functionality is based on the comparison of the threshold value with half of the difference of the acceleration of the current (x,y) sample and the previous one (refer to Section 3.2.1 Accelerometer slope filter).

At the very first sample, the slope filter output is calculated as half of the difference of the current sample [e.g. (x,y) = (0,1*g*)] with the previous one which is (x,y)=(0,0) since it doesn't exist. For this reason, on the y-axis the first output value of the slope filter is (1*g* - 0)/2=500 m*g* and it could be higher than the threshold value in which case a spurious interrupt event is generated. The interrupt signal is kept high for 1 ODR then it goes low.

In order to avoid this spurious interrupt generation, multiple solutions are possible. Hereafter are three alternative solutions (for the slope filter case):

a. Ignore the first generated wake-up signal;

b. Add a wait time higher than 1 ODR before driving the interrupt signal to the INT1/2 pin;

c. Initially set a higher ODR (833 Hz) so the first 2 samples are generated in a shorter period of time, reducing the slope filter latency time, then set the desired ODR (e.g. 12.5 Hz) and drive the interrupt signal on the pin, as indicated in the procedure below:

1. Write 00h to WAKE_UP_DUR // No duration and selection of wake-up threshold weight (1 LSB = FS_XL / 2^6)
2. Write 02h to WAKE_UP_THS // Set wake-up threshold
3. Write 51h to TAP_CFG0 // Enable interrupts and apply slope filter; latch mode disabled
4. Write 80h to TAP_CFG2 // Enable interrupt function
5. Write 7Ch to CTRL1_XL // Turn on the accelerometer
   // ODR_XL = 833 Hz, FS_XL = ±2 *g*
6. Wait 4 ms // Insert (reduced) wait time
7. Write 1Ch to CTRL1_XL // ODR_XL = 12.5 Hz
8. Write 20h to MD1_CFG // Wake-up interrupt driven to INT1 pin

5.3 Single-tap and double-tap recognition

The single-tap and double-tap recognition help to create a man-machine interface with little software loading. The device can be configured to output an interrupt signal on a dedicated pin when tapped in any direction.

If the sensor is exposed to a single input stimulus, it generates an interrupt request on the inertial interrupt pin INT1 and/or INT2. A more advanced feature allows the generation of an interrupt request when a double input stimulus with programmable time between the two events is recognized, enabling a mouse button-like function.

The single-tap and double-tap recognition functions use the slope between two consecutive acceleration samples to detect the tap events; the slope data is calculated using the following formula:

$$slope(t_n) = [\ acc(t_n) - acc(t_{n-1})\ ] / 2$$

This function can be fully programmed by the user in terms of expected amplitude and timing of the slope data by means of a dedicated set of registers.

Single and double-tap recognition work independently of the selected output data rate. Recommended minimum accelerometer ODR for these functions is 416 Hz.

In order to enable the single-tap and double-tap recognition functions it is necessary to set the INTERRUPTS_ENABLE bit in TAP_CFG2 register to 1.

5.3.1 Single tap

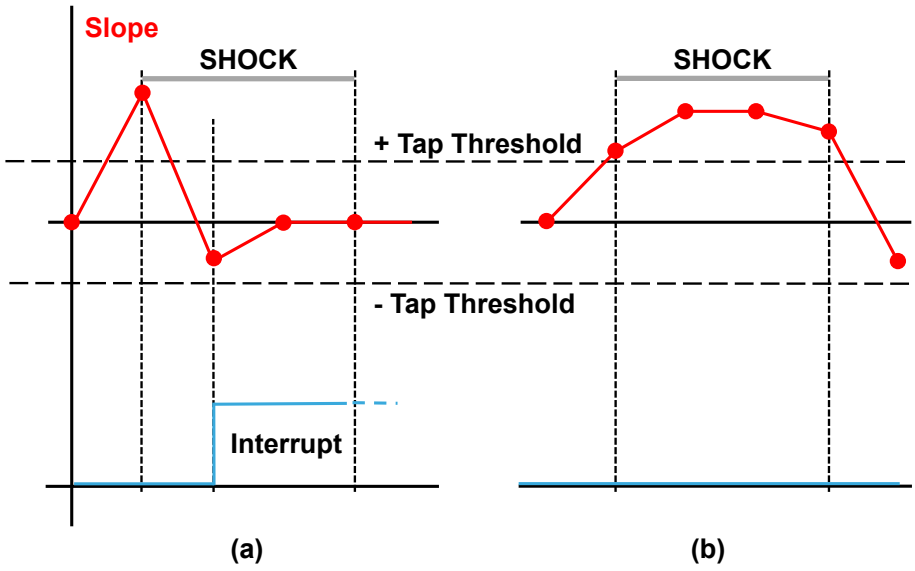
If the device is configured for single-tap event detection, an interrupt is generated when the slope data of the selected channel exceeds the programmed threshold, and returns below it within the Shock time window.

In the single-tap case, if the LIR bit of the TAP_CFG0 register is set to 0, the interrupt is kept active for the duration of the Quiet window. If the LIR bit is set to 1, the interrupt is kept active until the TAP_SRC or ALL_INT_SRC register is read.

The SINGLE_DOUBLE_TAP bit of WAKE_UP_THS has to be set to 0 in order to enable single-tap recognition only.

In case (a) of Figure 12. Single-tap event recognition the single-tap event has been recognized, while in case (b) the tap has not been recognized because the slope data falls below the threshold after the Shock time window has expired.

Figure 12. Single-tap event recognition



5.3.2 Double tap

If the device is configured for double-tap event detection, an interrupt is generated when, after a first tap, a second tap is recognized. The recognition of the second tap occurs only if the event satisfies the rules defined by the Shock, the Quiet and the Duration time windows.

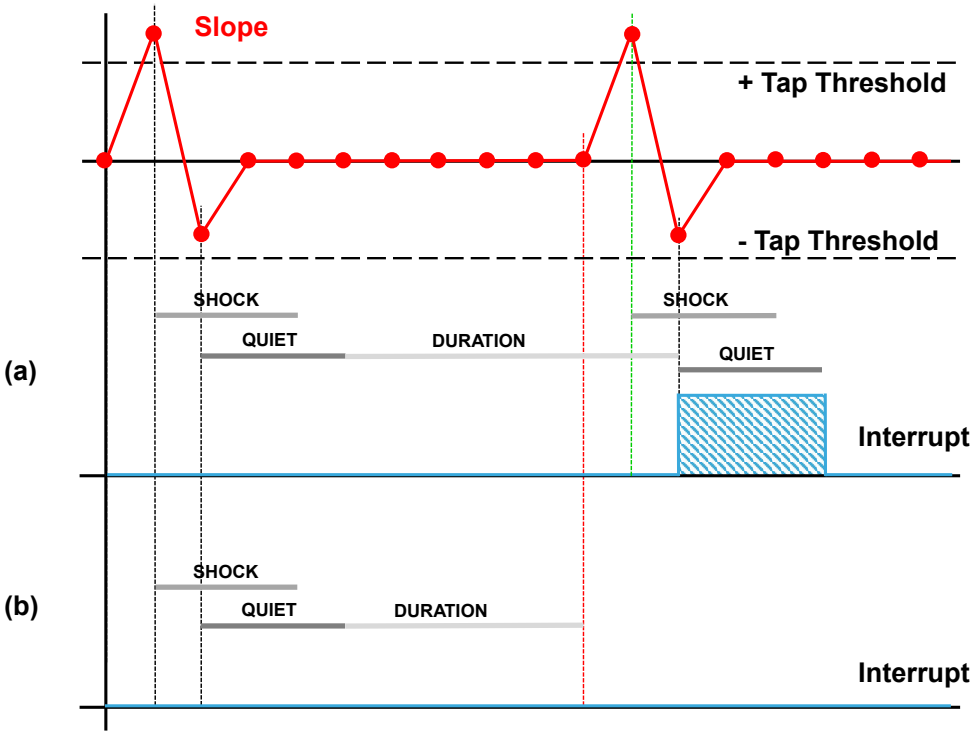
In particular, after the first tap has been recognized, the second tap detection procedure is delayed for an interval defined by the Quiet time. This means that after the first tap has been recognized, the second tap detection procedure starts only if the slope data exceeds the threshold after the Quiet window but before the Duration window has expired. In case (a) of Figure 13, a double-tap event has been correctly recognized, while in case (b) the interrupt has not been generated because the slope data exceeds the threshold after the window interval has expired.

Once the second tap detection procedure is initiated, the second tap is recognized with the same rule as the first: the slope data must return below the threshold before the Shock window has expired.

It is important to appropriately define the Quiet window to avoid unwanted taps due to spurious bouncing of the input signal.

In the double-tap case, if the LIR bit of the TAP_CFG0 register is set to 0, the interrupt is kept active for the duration of the Quiet window. If the LIR bit is set to 1, the interrupt is kept active until the TAP_SRC or ALL_INT_SRC register is read.

Figure 13. Double-tap event recognition (LIR bit = 0)



5.3.3 Single-tap and double-tap recognition configuration

The device can be configured to output an interrupt signal when tapped (once or twice) in any direction: the TAP_X_EN and TAP_Y_EN bits of the TAP_CFG0 register must be set to 1 to enable the tap recognition on the X, Y directions, respectively. In addition, the INTERRUPTS_ENABLE bit of the TAP_CFG2 register has to be set to 1.

Configurable parameters for tap recognition functionality are the tap thresholds (each axis has a dedicated threshold) and the Shock, Quiet and Duration time windows.

The TAP_THS_X[4:0] bits of the TAP_CFG1 register and the TAP_THS_Y[4:0] bits of the TAP_CFG2 register are used to select the unsigned threshold value used to detect the tap event on the respective axis. The value of 1 LSB of these 5 bits depends on the selected accelerometer full scale: 1 LSB = (FS_XL)/(2^5). The unsigned threshold is applied to both positive and negative slope data.

Note: If selecting FS_XL = ±3 g, 1 LSB = 4 / 2^5 g.

Both single-tap and double-tap recognition functions apply to only one axis. If more than one axis are enabled and they are over the respective threshold, the algorithm continues to evaluate only the axis with highest priority. The priority can be configured through the TAP_PRIORITY bit of TAP_CFG1. If the TAP_PRIORITY bit is set to 0, the X-axis has the priority over the Y-axis; otherwise, if the TAP_PRIORITY bit is set to 1, the Y-axis has the priority over the X-axis.

The Shock time window defines the maximum duration of the overcoming threshold event: the acceleration must return below the threshold before the Shock window has expired, otherwise the tap event is not detected. The SHOCK[1:0] bits of the INT_DUR2 register are used to set the Shock time window value: the default value of these bits is 00b and corresponds to 4/ODR_XL time, where ODR_XL is the accelerometer output data rate. If the SHOCK[1:0] bits are set to a different value, 1 LSB corresponds to 8/ODR_XL time.

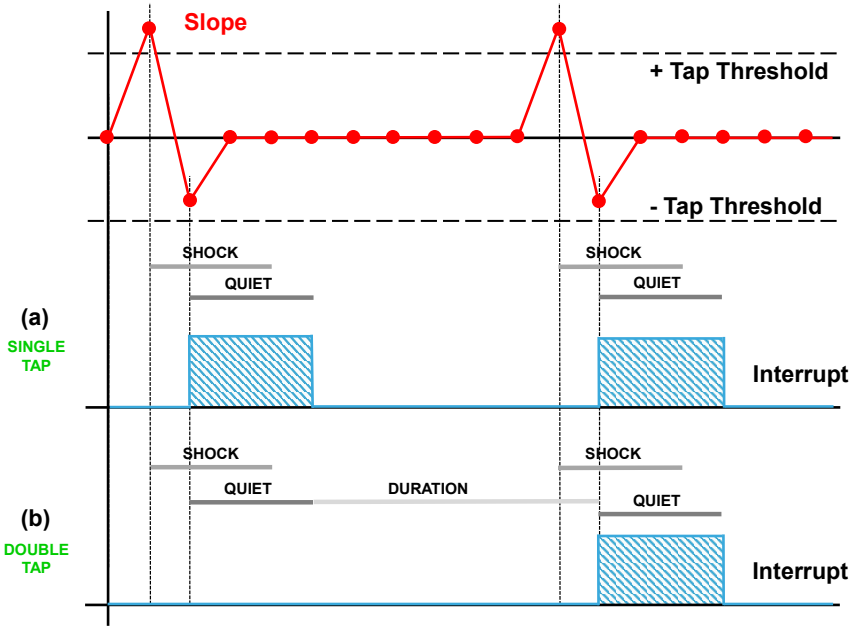
In the double-tap case, the Quiet time window defines the time after the first tap recognition in which there must not be any overcoming threshold event. When latched mode is disabled (LIR bit of TAP_CFG is set to 0), the Quiet time also defines the length of the interrupt pulse (in both single and double-tap case). The QUIET[1:0] bits of the INT_DUR2 register are used to set the Quiet time window value: the default value of these bits is 00b and corresponds to 2/ODR_XL time, where ODR_XL is the accelerometer output data rate. If the QUIET[1:0] bits are set to a different value, 1 LSB corresponds to 4/ODR_XL time.

In the double-tap case, the Duration time window defines the maximum time between two consecutive detected taps. The Duration time period starts just after the completion of the Quiet time of the first tap. The DUR[3:0] bits of the INT_DUR2 register are used to set the Duration time window value: the default value of these bits is 0000b and corresponds to 16/ODR_XL time, where ODR_XL is the accelerometer output data rate. If the DUR[3:0] bits are set to a different value, 1 LSB corresponds to 32/ODR_XL time.

Figure 14. Single and double-tap recognition (LIR bit = 0) illustrates a single-tap event (a) and a double-tap event (b). These interrupt signals can be driven to the two interrupt pins by setting to 1 the INT1_SINGLE_TAP bit of the MD1_CFG register or the INT2_SINGLE_TAP bit of the MD2_CFG register for the single-tap case, and setting to 1 the INT1_DOUBLE_TAP bit of the MD1_CFG register or the INT2_DOUBLE_TAP bit of the MD2_CFG register for the double-tap case.

No single/double-tap interrupt is generated if the accelerometer is in Stationary status (see Section 5.4 Motion/ Stationary recognition for more details).

Figure 14. **Single and double-tap recognition (LIR bit = 0)**



Tap interrupt signals can also be checked by reading the TAP_SRC (1Ch) register, described in the following table.

Table 14. **TAP_SRC register**

b7	b6	b5	b4	b3	b2	b1	b0
0	TAP_IA	SINGLE _TAP	DOUBLE _TAP	TAP_ SIGN	X_TAP	Y_TAP	0

- TAP_IA is set high when a single-tap or double-tap event has been detected.
- SINGLE_TAP is set high when a single tap has been detected.
- DOUBLE_TAP is set high when a double tap has been detected.
- TAP_SIGN indicates the acceleration sign when the tap event is detected. It is set low in case of positive sign and it is set high in case of negative sign.
- X_TAP/Y_TAP is set high when the tap event has been detected on the X/Y axis.

Single and double-tap recognition works independently. Setting the SINGLE_DOUBLE_TAP bit of the WAKE_UP_THS register to 0, only the single-tap recognition is enabled: double-tap recognition is disabled and cannot be detected. When the SINGLE_DOUBLE_TAP is set to 1, both single and double-tap recognition are enabled.

If latched mode is enabled and the interrupt signal is driven to the interrupt pins, the value assigned to SINGLE_DOUBLE_TAP also affects the behavior of the interrupt signal: when it is set to 0, the latched mode is applied to the single-tap interrupt signal; when it is set to 1, the latched mode is applied to the double-tap interrupt signal only. The latched interrupt signal is kept active until the TAP_SRC or ALL_INT_SRC register is read. The TAP_SIGN, X_TAP, Y_TAP bits are maintained at the state in which the interrupt was generated until the read is performed, and released at the next ODR cycle. In case the TAP_SIGN, X_TAP, Y_TAP bits have to be evaluated (in addition to the TAP_IA bit), it is recommended to directly read the TAP_SRC register (do not use ALL_INT_SRC register for this specific case). If latched mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

5.3.4 Single-tap example

A basic SW routine for single-tap detection is given below.

1.	Write 6Ch to CTRL1_XL	// Turn on the accelerometer
		// ODR_XL = 416 Hz, FS_XL = ±2 *g*
2.	Write 0Ch to TAP_CFG0	// Enable tap detection on X and Y axes
3.	Write 09h to TAP_CFG1	// Set X-axis threshold and axes priority
4.	Write 89h to TAP_CFG2	// Set Y-axis threshold and enable interrupt
5.	Write 06h to INT_DUR2	// Set Quiet and Shock time windows
6.	Write 00h to WAKE_UP_THS	// Only single-tap enabled (SINGLE_DOUBLE_TAP = 0)
7.	Write 40h to MD1_CFG	// Single-tap interrupt driven to INT1 pin

In this example the TAP_THS_X[4:0] and TAP_THS_Y[4:0] bits are set to 01001b, therefore the tap threshold for each axis is 562.5 m*g* (= 9 * FS_XL / 2^5).

The SHOCK field of the INT_DUR2 register is set to 10b: an interrupt is generated when the slope data exceeds the programmed threshold, and returns below it within 38.5 ms (= 2 * 8 / ODR_XL) corresponding to the Shock time window.

The QUIET field of the INT_DUR2 register is set to 01b: since latched mode is disabled, the interrupt is kept high for the duration of the Quiet window, therefore 9.6 ms (= 1 * 4 / ODR_XL).

5.3.5 Double-tap example

A basic SW routine for double-tap detection is given below.

1.	Write 6Ch to CTRL1_XL	// Turn on the accelerometer
		// ODR_XL = 416 Hz, FS_XL = ±2 *g*
2.	Write 0Ch to TAP_CFG0	// Enable tap detection on X and Y axes
3.	Write 0Ch to TAP_CFG1	// Set X-axis threshold and axes priority
4.	Write 8Ch to TAP_CFG2	// Set Y-axis threshold and enable interrupt
5.	Write 7Fh to INT_DUR2	// Set Duration, Quiet and Shock time windows
6.	Write 80h to WAKE_UP_THS	// Single-tap and double-tap enabled (SINGLE_DOUBLE_TAP = 1)
7.	Write 08h to MD1_CFG	// Double-tap interrupt driven to INT1 pin

In this example the TAP_THS_X[4:0] and TAP_THS_Y[4:0] bits are set to 01100b, therefore the tap threshold is 750 m*g* (= 12 * FS_XL / 2^5).

For interrupt generation, during the first and the second tap the slope data must return below the threshold before the Shock window has expired. The SHOCK field of the INT_DUR2 register is set to 11b, therefore the Shock time is 57.7 ms (= 3 * 8 / ODR_XL).

For interrupt generation, after the first tap recognition there must not be any slope data overthreshold during the Quiet time window. Furthermore, since latched mode is disabled, the interrupt is kept high for the duration of the Quiet window. The QUIET field of the INT_DUR2 register is set to 11b, therefore the Quiet time is 28.8 ms (= 3 * 4 / ODR_XL).

For the maximum time between two consecutive detected taps, the DUR field of the INT_DUR2 register is set to 0111b, therefore the Duration time is 538.5 ms (= 7 * 32 / ODR_XL).

5.4 Motion/Stationary recognition

The working principle of the Motion/Stationary embedded function is similar to wake-up. If no movement condition is detected for a programmable time, a Stationary condition event is generated; otherwise, when the accelerometer data exceed the configurable threshold, a Motion condition event is generated.

The Motion/Stationary recognition function is enabled by setting the INTERRUPTS_ENABLE bit to 1.

The Motion/Stationary recognition function can be implemented using either the slope filter (see Section 3.2.1 Accelerometer slope filter for more details) or the high-pass digital filter, as illustrated in Figure 2. Accelerometer filtering chain. The filter to be applied can be selected using the SLOPE_FDS bit of the TAP_CFG0 register: if this bit is set to 0 (default value), the slope filter is used; if it is set to 1, the high-pass digital filter is used.

This function can be fully programmed by the user in terms of expected amplitude and timing of the filtered data by means of a dedicated set of registers (Figure 15. Motion/Stationary recognition (using the slope filter)).

The unsigned threshold value is defined using the WK_THS[5:0] bits of the WAKE_UP_THS register; the value of 1 LSB of these 6 bits depends on the selected accelerometer full scale and on the value of the WAKE_THS_W bit of the WAKE_UP_DUR register:

- if WAKE_THS_W = 0, 1 LSB = FS_XL / 2^6;
- if WAKE_THS_W = 1, 1 LSB = FS_XL / 2^8.

Note: If selecting FS_XL = ±3 g:

- *If WAKE_THS_W = 0, 1 LSB = 4 / 2^6 g;*
- *If WAKE_THS_W = 1, 1 LSB = 4 / 2^8 g.*

The threshold is applied to both positive and negative filtered data.

The duration of the Stationary status to be recognized is defined by the SLEEP_DUR[3:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to 512/ODR_XL time, where ODR_XL is the accelerometer output data rate. If the SLEEP_DUR[3:0] bits are set to 0000b, the duration of the Stationary status to be recognized is equal to 16 / ODR_XL time.

When the Stationary status is detected, the interrupt is set high for 1/ODR_XL[s] period then it is automatically deasserted.

When filtered data on one axis becomes bigger than the threshold for a configurable time, Motion status is detected. The duration of the Motion status to be recognized is defined by the WAKE_DUR[1:0] bits of the WAKE_UP_DUR register. 1 LSB corresponds to 1 / ODR_XL time, where ODR_XL is the accelerometer output data rate.

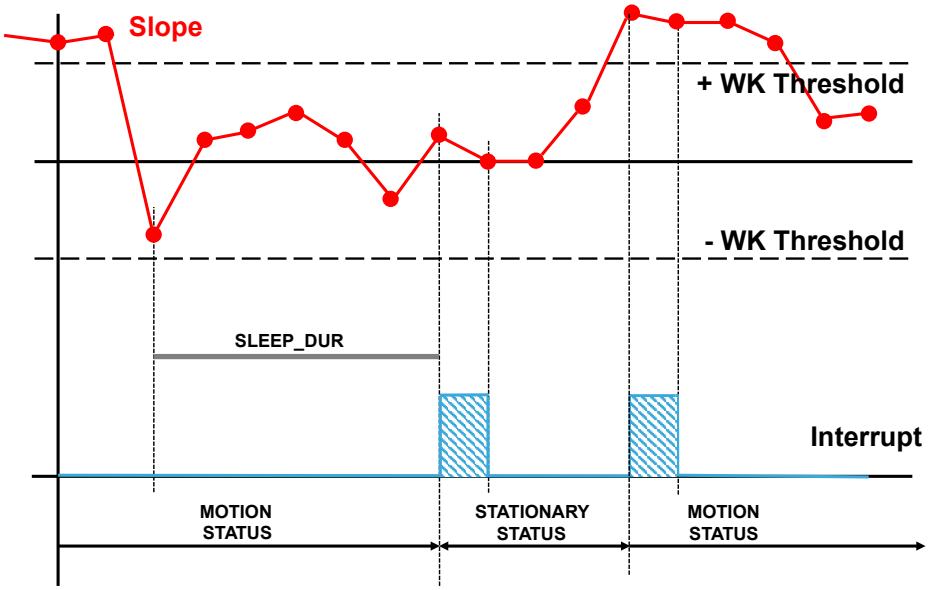
When the Motion status is detected, the interrupt is set high for 1/ODR_XL[s] period then it is automatically deasserted.

Once the Motion/Stationary detection function is enabled, the status can be driven to the two interrupt pins by setting to 1 the INT1_SLEEP_CHANGE bit of the MD1_CFG register or the INT2_SLEEP_CHANGE bit of the MD2_CFG register; it can also be checked by reading the SLEEP_CHANGE_IA bit of the WAKE_UP_SRC or ALL_INT_SRC register.

The SLEEP_CHANGE_IA bit is by default in pulsed mode. Latched mode can be selected by setting the LIR bit of the TAP_CFG0 register to 1 and the INT1_SLEEP_CHANGE bit of the MD1_CFG register or the INT2_SLEEP_CHANGE bit of the MD2_CFG register to 1. The SLEEP_STATE bit of the WAKE_UP_SRC register is not affected by the LIR configuration: it corresponds to the current state of the device when the WAKE_UP_SRC register is read.

By setting the SLEEP_STATUS_ON_INT bit of the TAP_CFG0 register to 1, the signal routed to the INT1 or INT2 pins is configured to be the Motion/Stationary state (SLEEP_STATE bit of WAKE_UP_SRC register) instead of the sleep-change signal: it goes high during Stationary state and it goes low during Motion state. Latched mode is not supported in this configuration.

Figure 15. **Motion/Stationary recognition (using the slope filter)**



A basic SW routine for Motion/Stationary detection is as follows:

1.	Write 5Ch to CTRL1_XL	// Turn on the accelerometer
		// ODR_XL = 208 Hz, FS_XL = ±2 *g*
2.	Write 02h to WAKE_UP_DUR	// Set duration for Stationary detection
		// Select Motion/Stationary threshold resolution and duration
3.	Write 02h to WAKE_UP_THS	// Set Motion/Stationary threshold
4.	Write 00h to TAP_CFG0	// Select sleep-change notification
		// Select slope filter
5.	Write 80h to TAP_CFG2	// Enable interrupt
6.	Write 80h to MD1_CFG	// Motion/Stationary interrupt driven to INT1 pin

In this example the WK_THS field of the WAKE_UP_THS register is set to 000010b, therefore the Motion/Stationary threshold is 62.5 m*g* (= 2 * FS_XL / 2^6 since the WAKE_THS_W bit of the WAKE_UP_DUR register is set to 0).

Before Stationary detection, the X,Y slope data must be smaller than the configured threshold for a period of time defined by the SLEEP_DUR field of the WAKE_UP_DUR register: this field is set to 0010b, corresponding to 4.92 s (= 2 * 512 / ODR_XL). After this period of time has elapsed, the Stationary status is detected.

The Motion status is detected as soon as the slope data of (at least) one axis are bigger than the threshold for one sample, since the WAKE_DUR[1:0] bits of the WAKE_UP_DUR register are configured to 00b.

5.5 Boot status

After the device is powered up, it performs a 10 ms (maximum) boot procedure to load the trimming parameters. After the boot is completed, the accelerometer is automatically configured in Power-Down mode. During the boot time the registers are not accessible.

After power up, the trimming parameters can be re-loaded by setting the BOOT bit of the CTRL3_C register to 1.

No toggle of the device power lines is required and the content of the device control registers is not modified, so the device operating mode doesn't change after boot. If the reset to the default value of the control registers is required, it can be performed by setting the SW_RESET bit of the CTRL3_C register to 1. When this bit is set to 1, the following registers are reset to their default value:

- FUNC_CFG_ACCESS (01h);
- PIN_CTRL (02h);
- FIFO_CTRL1 (07h) through FIFO_CTRL4 (0Ah);
- COUNTER_BDR_REG1 (0Bh) and COUNTER_BDR_REG2 (0Ch);
- INT1_CTRL (0Dh) and INT2_CTRL (0Eh);
- CTRL1_XL (10h) through CTRL10_C (19h);
- FIFO_STATUS1 (3Ah) and FIFO_STATUS2 (3Bh);
- TAP_CFG0 (56h) through MD2_CFG (5Fh);
- X_OFS_USR (73h) and Y_OFS_USR (74h).

The SW_RESET procedure can take 50 µs; the status of reset is signaled by the status of the SW_RESET bit of the CTRL3_C register: once the reset is completed, this bit is automatically set low.

The boot status signal is driven to the INT1 interrupt pin by setting the INT1_BOOT bit of the INT1_CTRL register to 1: this signal is set high while the boot is running and it is set low again at the end of the boot procedure.

The reboot flow is as follows:

1. Set the accelerometer in Power-Down mode;
2. Set INT1_BOOT bit of INT1_CTRL register to 1 [optional];
3. Set BOOT bit of CTRL3_C register to 1;
4. Monitor reboot status, three possibilities:
   a. Wait 10 ms;
   b. Monitor INT1 pin until it returns to 0 (step 2. is mandatory in this case);
   c. Poll BOOT bit of CTRL3_C until it returns to 0.

Reset flow is as follows:

1. Set the accelerometer in Power-down mode;
2. Set to 1 the SW_RESET bit of CTRL3_C to 1;
3. Monitor software reset status, two possibilities:
   a. Wait 50 µs
   b. Poll SW_RESET bit of CTRL3_C until it returns to 0.

In order to avoid conflicts, the reboot and the sw reset must not be executed at the same time (do not set to 1 at the same time both the BOOT bit and SW_RESET bit of CTRL3_C register). The above flows must be performed serially.

6 Timestamp

Together with sensor data the device can provide timestamp information.

To enable this functionality the TIMESTAMP_EN bit of the CTRL10_C register has to be set to 1. The time step count is given by the concatenation of the TIMESTAMP3 & TIMESTAMP2 & TIMESTAMP1 & TIMESTAMP0 registers and is represented as a 32-bit unsigned number.

The nominal timestamp resolution is 25 µs. It is possible to get the actual timestamp resolution value through the FREQ_FINE[7:0] bits of the INTERNAL_FREQ_FINE register, which contains the difference in percentage of the actual ODR (and timestamp rate) with respect to the nominal value.

$$t_{actual}[s] = \frac{1}{40000 \cdot (1 + 0.0015 \cdot FREQ\_FINE)}$$

Similarly, it is possible to get the actual output data rate by using the following formula:

$$ODR_{actual}[Hz] = \frac{6667 + 0.0015 \cdot FREQ\_FINE \cdot 6667}{ODR_{coeff}}$$

where the ODR_{coeff} values are indicated in the table below.

Table 15. ODR_{coeff} values

Selected ODR [Hz]	ODR_{coeff}
12.5	512
26	256
52	128
104	64
208	32
416	16
833	8

If the accelerometer is in Power-Down mode, the timestamp counter does not work and the timestamp value is frozen at the last value.

When the maximum value 4294967295 LSB (equal to FFFFFFFFh) is reached corresponding to approximately 30 hours, the counter is automatically reset to 00000000h and continues to count. The timer count can be reset to zero at any time by writing the reset value AAh in the TIMESTAMP2 register.

The TIMESTAMP_ENDCOUNT bit of the ALL_INT_SRC goes high 6.4 ms before the occurrence of a timestamp overrun condition. This flag is reset when the ALL_INT_SRC register is read. It is also possible to route this signal on the INT2 pin (75 µs duration pulse) by setting the INT2_TIMESTAMP bit of MD2_CFG to 1.

The timestamp can be batched in FIFO (see Section 8 First-in, first-out (FIFO) buffer for details).

7 Mode 2 - Sensor hub mode

The hardware flexibility of the IIS2ICLX allows connecting the pins with different mode connections to external sensors to expand functionalities such as adding a sensor hub. When sensor hub mode (Mode 2) is enabled, both the primary I²C/SPI (3- and 4-wire) slave interface and the I²C master interface for the connection of external sensors are available. Mode 2 connection mode is described in detail in the following paragraphs.

7.1 Sensor hub mode description

In sensor hub mode (Mode 2) up to 4 external sensors can be connected to the I²C master interface of the device. The sensor hub trigger signal can be synchronized with the accelerometer data-ready signal (up to 104 Hz). In this configuration, the sensor hub ODR can be configured through the SHUB_ODR_[1:0] bits of the SLAVE0_CONFIG register. Alternatively, an external signal connected to the INT2 pin can be used as the sensor hub trigger. In this second case, the maximum ODR supported for external sensors depends on the number of read / write operations that can be executed between two consecutive trigger signals.

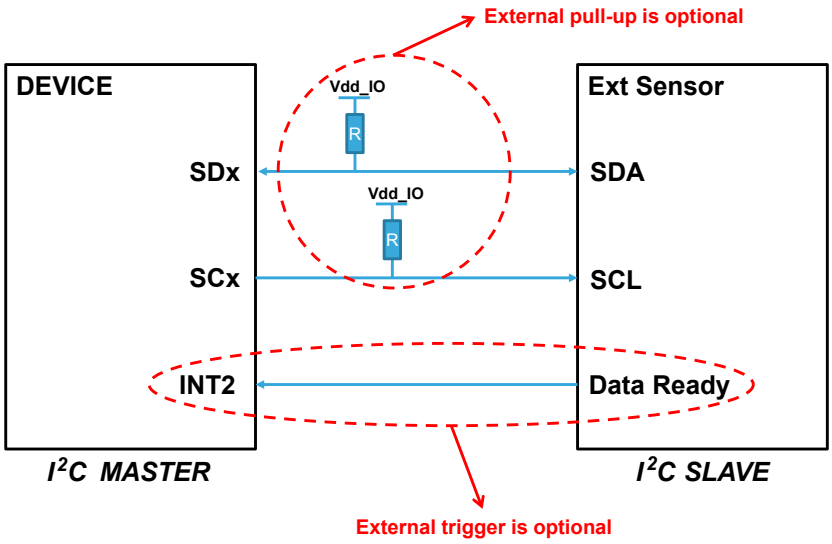
On the sensor hub trigger signal, all the write and read I²C operations configured through the registers SLVx_ADD, SLVx_SUBADD, SLAVEx_CONFIG and DATAWRITE_SLV0 are performed sequentially from external sensor 0 to external sensor 3 (depending on the external sensors enabled through the AUX_SENS_ON[1:0] field in the MASTER_CONFIG register).

External sensor data can also be stored in FIFO (see Section 8 First-in, first-out (FIFO) buffer for details).

If the accelerometer is in Power-Down mode, the sensor hub does not work.

All external sensors have to be connected in parallel to the SDx/SCx pins of the device, as illustrated in Figure 16. External sensor connections in Mode 2 for a single external sensor. External pull-up resistors and the external trigger signal connection are optional and depend on the configuration of the registers.

Figure 16. External sensor connections in Mode 2



7.2 Sensor hub mode registers

The sensor hub configuration registers and output registers are accessible when the bit SHUB_REG_ACCESS of the FUNC_CFG_ACCESS register is set to 1. After setting the SHUB_REG_ACESS bit to 1, only sensor hub registers are available. In order to guarantee the correct register mapping for other operations, after the sensor hub configuration or output data reading, the SHUB_REG_ACCESS bit of the FUNC_CFG_ACCESS register must be set to 0.

The MASTER_CONFIG register has to be used for the configuration of the I²C master interface.

A set of registers SLVx_ADD, SLVx_SUBADD, SLAVEx_CONFIG is dedicated to the configuration of the 4 slave interfaces associated to the 4 connectable external sensors. An additional register, DATAWRITE_SLV0, is associated to slave #0 only. It has to be used to implement the write operations.

Finally, 18 registers (from SENSOR_HUB_1 to SENSOR_HUB_18) are available to store the data read from the external sensors.

7.2.1 MASTER_CONFIG (14h)

This register is used to configure the I²C master interface.

Table 16. MASTER_CONFIG register

b7	b6	b5	b4	b3	b2	b1	b0
RST_ MASTER_REGS	WRITE_ ONCE	START_ CONFIG	PASS_THROUGH _MODE	SHUB_ PU_EN	MASTER_ON	AUX_ SENS_ON1	AUX_ SENS_ON0

- RST_MASTER_REGS bit is used to reset the I²C master interface, configuration and output registers. It must be manually asserted and de-asserted.
- WRITE_ONCE bit is used to limit the write operations on slave 0 to only one occurrence (avoiding to repeat the same write operation multiple times). If this bit is not asserted, a write operation is triggered at each ODR.

Note: The WRITE_ONCE bit must be set to 1 if the slave 0 is used for reading.

- START_CONFIG bit selects the sensor hub trigger signal.
  - When this bit is set to 0, the accelerometer sensor has to be active (not in Power-Down mode) and the sensor hub trigger signal is the accelerometer data-ready signal, with a frequency defined by the SHUB_ODR_[1:0] bits of the SLAVE0_CONFIG register (up to 104 Hz).
  - When this bit is set to 1, the accelerometer has to be active and the sensor hub trigger signal is the INT2 pin. In fact, when both the MASTER_ON bit and START_CONFIG bit are set to 1, the INT2 pin is configured as an input signal. In this case, the INT2 pin has to be connected to the data-ready pin of the external sensor (Figure 16. External sensor connections in Mode 2) in order to trigger the read/ write operations on the external sensor registers. Sensor hub interrupt from INT2 is ‘high-level triggered’ (not programmable).

  Note: In case of external trigger signal usage (START_CONFIG=1), if the INT2 pin is connected to the data-ready pin of the external sensor (Figure 16. External sensor connections in Mode 2) and the latter is in Power-Down mode, then no data-ready signal can be generated by the external sensor. For this reason, the initial configuration of the external sensor’s register has to be performed using the internal trigger signal (START_CONFIG=0). After the external sensor is activated and the data-ready signal is available, the external trigger signal can be used by switching the START_CONFIG bit to 1.
- PASS_THROUGH_MODE bit is used to enable/disable the I²C interface pass-through. When this bit is set to 1, the main I²C line (e.g. connected to an external microcontroller) is short-circuited with the auxiliary one, in order to implement a direct access to the external sensor registers. See Section 7.3 Sensor hub pass-through feature for details.
- SHUB_PU_EN bit enables/disables the internal pull-up on the I²C master line. When this bit is set to 0, the internal pull-up is disabled and the external pull-up resistors on the SDx/SCx pins are required, as shown in Figure 16. External sensor connections in Mode 2. When this bit is set to 1, the internal pull-up is enabled (regardless of the configuration of the MASTER_ON bit) and the external pull-up resistors on the SDx/SCx pins are not required.

- MASTER_ON bit has to be set to 1 to enable the auxiliary I²C master of the device (sensor hub mode). In order to change the sensor hub configuration at runtime or when setting the accelerometer sensor in Power-Down mode, or when applying the software reset procedure, the I²C master must be disabled, followed by a 300 µs wait. The following procedure must be implemented:

1. Turn off I²C master by setting MASTER_ON = 0.
2. Wait 300 µs.
3. Change the configuration of the sensor hub registers or set the accelerometer in Power-Down mode or apply the software reset procedure.

- AUX_SENS_ON[1:0] bits have to be set accordingly to the number of slaves to be used. I²C transactions are performed sequentially from slave 0 to slave 3. The possible values are:
  - 00b: one slave;
  - 01b: two slaves;
  - 10b: three slaves;
  - 11b: four slaves.

7.2.2 STATUS_MASTER (22h)

The STATUS_MASTER register, similarly to the other sensor hub configurations and output registers, can be read only after setting the SHUB_REG_ACCESS bit of the FUNC_CFG_ACCESS register to 1. The STATUS_MASTER register is also mapped to the STATUS_MASTER_MAINPAGE register, which can be directly read without enabling access to the sensor hub registers.

Table 17. STATUS_MASTER / STATUS_MASTER_MAINPAGE register

b7	b6	b5	b4	b3	b2	b1	b0
WR_ONCE_DONE	SLAVE3_NACK	SLAVE2_NACK	SLAVE1_NACK	SLAVE0_NACK	0	0	SENS_HUB_ENDOP

- WR_ONCE_DONE bit is set to 1 after a write operation performed with the WRITE_ONCE bit configured to 1 in the MASTER_CONFIG register. This bit can be polled in order to check if the single write transaction has been completed.
- SLAVEx_NACK bits are set to 1 if a “not acknowledge” event happens during the communication with the corresponding slave x.
- SENS_HUB_ENDOP bit reports the status of the I²C master: during the idle state of the I²C master, this bit is equal to 1; it goes to 0 during I²C master read/write operations.

When a sensor hub routine is completed, this bit automatically goes to 1 and the external sensor data are available to be read from the SENSOR_HUB_x registers (depending on the configuration of the SLVx_ADD, SLVx_SUBADD, SLAVEx_CONFIG registers).

Information about the status of the I²C master can be driven to the INT1 interrupt pin by setting the INT1_SHUB bit of the MD1_CFG register to 1.This signal goes high on a rising edge of the SENS_HUB_ENDOP signal and it is cleared only if the STATUS_MASTER / STATUS_MASTER_MAINPAGE register is read.

7.2.3 SLV0_ADD (15h), SLV0_SUBADD (16h), SLAV0_CONFIG (17h)

The sensor hub registers used to configure the I²C slave interface associated to the first external sensor are described hereafter.

Table 18. SLV0_ADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave0 _add6	slave0 _add5	slave0 _add4	slave0 _add3	slave0 _add2	slave0 _add1	slave0 _add0	rw_0

- slave0_add[6:0] bits are used to indicate the I²C slave address of the first external sensor.
- rw_0 bit configures the read/write operation to be performed on the first external sensor (0: write operation; 1: read operation). The read/write operation is executed when the next sensor hub trigger event occurs.

Table 19. SLV0_SUBADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave0 _reg7	slave0 _reg6	slave0 _reg5	slave0 _reg4	slave0 _reg3	slave0 _reg2	slave0 _reg1	slave0 _reg0

- slave0_reg[7:0] bits are used to indicate the address of the register of the first external sensor to be written (if the rw_0 bit of the SLV0_ADD register is set to 0) or the address of the first register to be read (if the rw_0 bit is set to 1).

Table 20. SLAVE0_CONFIG register

b7	b6	b5	b4	b3	b2	b1	b0
SHUB_ ODR_1	SHUB_ ODR_0	0	0	BATCH_EXT_ SENS_0_EN	Slave0 _numop2	Slave0 _numop1	Slave0 _numop0

- SHUB_ODR_[1:0] bits are used to configure the sensor hub output data rate when using internal trigger (accelerometer data-ready signal). The sensor hub output data rate can be configured to four possible values, limited by the ODR of the accelerometer sensor:
  - 00b: 104 Hz;
  - 01b: 52 Hz;
  - 10b: 26 Hz;
  - 11b: 12.5 Hz.

The maximum allowed value for the SHUB_ODR_[1:0] bits corresponds to the accelerometer selected ODR.

- BATCH_EXT_SENS_0_EN bit is used enable the batching in FIFO of the external sensor associated to slave0.
- Slave0_numop[2:0] bits are dedicated to define the number of consecutive read operations to be performed on the first external sensor starting from the register address indicated in the SLV0_SUBADD register.

7.2.4 SLV1_ADD (18h), SLV1_SUBADD (19h), SLAVE1_CONFIG (1Ah)

The sensor hub registers used to configure the I²C slave interface associated to the second external sensor are described hereafter.

Table 21. SLV1_ADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave1 _add6	slave1 _add5	slave1 _add4	slave1 _add3	slave1 _add2	slave1 _add1	slave1 _add0	r_1

- slave1_add[6:0] bits are used to indicate the I²C slave address of the second external sensor.
- r_1 bit enables/disables the read operation to be performed on the second external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

Table 22. SLV1_SUBADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave1 _reg7	slave1 _reg6	slave1 _reg5	slave1 _reg4	slave1 _reg3	slave1 _reg2	slave1 _reg1	slave1 _reg0

- Slave1_reg[7:0] bits are used to indicate the address of the register of the second external sensor to be read when the r_1 bit of SLV1_ADD register is set to 1.

Table 23. SLAVE1_CONFIG register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	BATCH_EXT_ SENS_1_EN	Slave1 _numop2	Slave1 _numop1	Slave1 _numop0

- BATCH_EXT_SENS_1_EN bit is used enable the batching in FIFO of the external sensor associated to slave1.

Slave1_numop[2:0] bits are dedicated to define the number of consecutive read operations to be performed on the second external sensor starting from the register address indicated in the SLV1_SUBADD register.

7.2.5 SLV2_ADD (1Bh), SLV2_SUBADD (1Ch), SLAVE2_CONFIG (1Dh)

The sensor hub registers used to configure the I²C slave interface associated to the third external sensor are described hereafter.

Table 24. SLV2_ADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave2 _add6	slave2 _add5	slave2 _add4	slave2 _add3	slave2 _add2	slave2 _add1	slave2 _add0	r_2

- Slave2_add[6:0] bits are used to indicate the I²C slave address of the third external sensor.
- r_2 bit enables/disables the read operation to be performed on the third external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

Table 25. SLV2_SUBADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave2 _reg7	slave2 _reg6	slave2 _reg5	slave2 _reg4	slave2 _reg3	slave2 _reg2	slave2 _reg1	slave2 _reg0

- Slave2_reg[7:0] bits are used to indicate the address of the register of the third external sensor to be read when the r_2 bit of the SLV2_ADD register is set to 1.

Table 26. SLAVE2_CONFIG register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	BATCH_EXT_ SENS_2_EN	Slave2 _numop2	Slave2 _numop1	Slave2 _numop0

- BATCH_EXT_SENS_2_EN bit is used enable the batching in FIFO of the external sensor associated to slave2.
- Slave2_numop[2:0] bits are dedicated to define the number of consecutive read operations to be performed on the third external sensor starting from the register address indicated in the SLV2_SUBADD register.

7.2.6 SLV3_ADD (1Eh), SLV3_SUBADD (1Fh), SLAVE3_CONFIG (20h)

The sensor hub registers used to configure the I²C slave interface associated to the fourth external sensor are described hereafter.

Table 27. SLV3_ADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave3 _add6	slave3 _add5	slave3 _add4	slave3 _add3	slave3 _add2	slave3 _add1	slave3 _add0	r_3

- Slave3_add[6:0] bits are used to indicate the I²C slave address of the fourth external sensor.
- r_3 bit enables/disables the read operation to be performed on the fourth external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

Table 28. SLV3_SUBADD register

b7	b6	b5	b4	b3	b2	b1	b0
slave3 _reg7	slave3 _reg6	slave3 _reg5	slave3 _reg4	slave3 _reg3	slave3 _reg2	slave3 _reg1	slave3 _reg0

- Slave3_reg[7:0] bits are used to indicate the address of the register of the fourth external sensor to be read when the r_3 bit of the SLV3_ADD register is set to 1.

Table 29. SLAVE3_CONFIG register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	BATCH_EXT_ SENS_3_EN	Slave3 _numop2	Slave3 _numop1	Slave3 _numop0

- BATCH_EXT_SENS_3_EN bit is used enable the batching in FIFO of the external sensor associated to slave3.

Slave3_numop[2:0] bits are dedicated to define the number of consecutive read operations to be performed on the fourth external sensor starting from the register address indicated in the SLV3_SUBADD register.

7.2.7 DATAWRITE_SLV0 (21h)

Table 30. DATAWRITE_SLV0 register

b7	b6	b5	b4	b3	b2	b1	b0
Slave0_ dataw7	Slave0_ dataw6	Slave0_ dataw5	Slave0_ dataw4	Slave0_ dataw3	Slave0_ dataw2	Slave0_ dataw1	Slave0_ dataw0

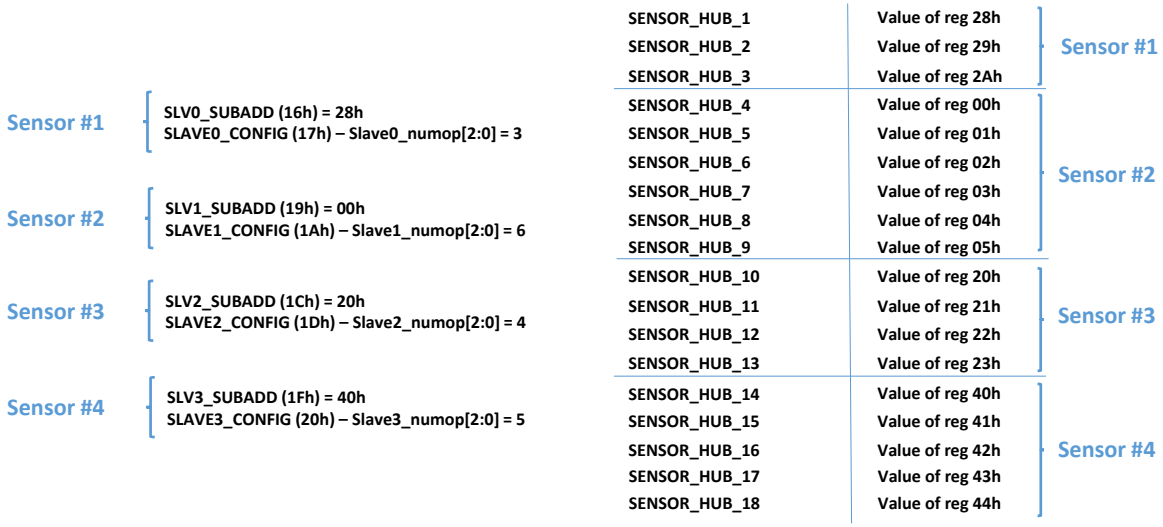
- Slave0_dataw[7:0] bits are dedicated, when the rw_0 bit of SLV0_ADD register is set to 0 (write operation), to indicate the data to be written to the first external sensor at the address specified in the SLV0_SUBADD register.

7.2.8 SENSOR_HUB_x registers

Once the auxiliary I²C master is enabled, for each of the external sensors it reads a number of registers equal to the value of the Slavex_numop (x = 0, 1, 2, 3) field, starting from the register address specified in the SLVx_SUBADD (x = 0, 1, 2, 3) register. The number of external sensors to be managed is specified in the AUX_SENS_ON[1:0] bits of the MASTER_CONFIG register.

Read data are consecutively stored (in the same order they are read) in the device registers starting from the SENSOR_HUB_1 register, as in the example in Figure 17. SENSOR_HUB_X allocation example; 18 registers, from SENSOR_HUB_1 to SENSOR_HUB_18, are available to store the data read from the external sensors.

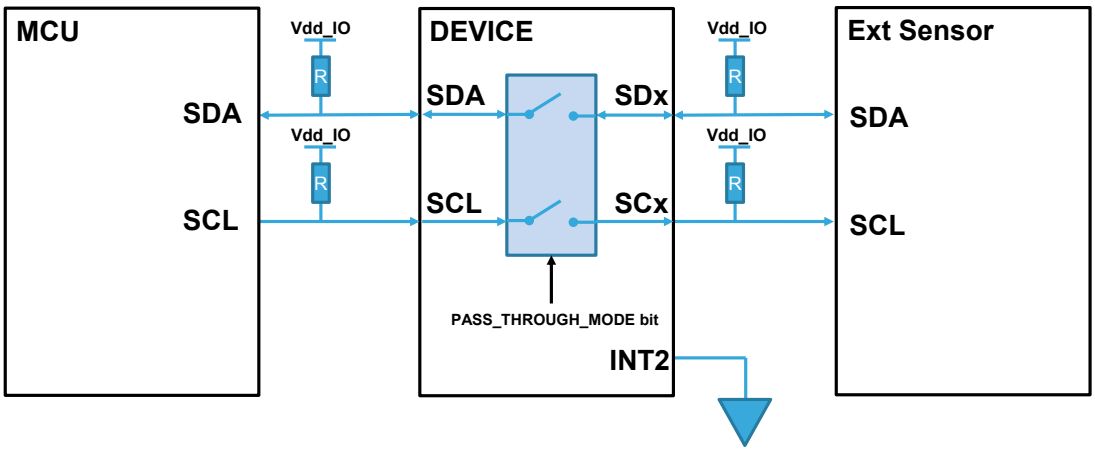
Figure 17. SENSOR_HUB_X allocation example



7.3 Sensor hub pass-through feature

The PASS_THROUGH_MODE bit of the MASTER_CONFIG register is used to enable/disable the I²C interface pass-through: when it is set to 1, the main I²C line (e.g. connected to an external microcontroller) is short-circuited with the auxiliary one in order to implement a direct access to the external sensor registers. The pass-through feature for external device configuration can be used only if I²C protocol is used on primary interface. This feature can be used to configure the external sensors.

Figure 18. Pass-through feature



Some limitations must be considered when using the sensor hub and the pass-through feature. Three different scenarios are possible:

1. The sensor hub is used with the START_CONFIG bit of the MASTER_CONFIG register set to 0 (internal trigger) and the pass-through feature is not used. There is no limitation on INT2 pin usage.
2. The sensor hub is used with the START_CONFIG bit of the MASTER_CONFIG register set to 0 (internal trigger) and the pass-through feature is used. The INT2 pin must be connected to GND. It is not possible to switch to external trigger configuration (by setting the START_CONFIG bit to 1) and the INT2 pin cannot be used for the digital interrupts. Specific procedures have to be applied to enable/disable the pass-through feature which are described in Section 7.3.1 Pass-through feature enable and in Section 7.3.2 Pass-through feature disable.
3. The sensor hub is used with the START_CONFIG bit of the MASTER_CONFIG register set to 1 (external trigger). The pass-through feature cannot be used. The INT2 pin has to be connected to the data-ready pin of the external sensor (trigger signal) and the procedure below has to be executed to avoid conflicts with the INT2 line:
   a. Set either the TRIG_EN or LVL1_EN or LVL2_EN bit of the CTRL6_C register to 1 (to configure the INT2 pin as input pin);
   b. Configure the external sensors (do not use the pass-through);
   c. Configure the sensor hub SLAVEx registers;
   d. Set the START_CONFIG bit of the MASTER_CONFIG register to 1;
   e. Set the MASTER_ON bit of the MASTER_CONFIG register to 1;
   f. Reset to 0 the bit in the CTRL6_C register asserted in step a.

Examples of external sensors configuration without using the pass-through is given in Section 7.4 Sensor hub mode example.

7.3.1 Pass-through feature enable

When the embedded sensor hub functionality is disabled, the pass-through feature can be enabled at any time by setting the PASS_THROUGH_MODE bit of the MASTER_CONFIG register to 1.

When the embedded sensor hub functionality is enabled, a specific procedure has to be followed to enable the pass-through feature in order to prevent I²C bus arbitration loss:

1. Set the START_CONFIG bit of the MASTER_CONFIG register to 1 in order to disable the sensor hub trigger (external trigger is enabled, but no trigger can be received on the INT2 pin since it's connected to GND);
2. Wait at least 5 ms (running I²C operations will be completed);
3. Set the MASTER_ON bit of the MASTER_CONFIG register to 0 in order to disable the embedded sensor hub;
4. Set the START_CONFIG bit of the MASTER_CONFIG register to 0 in order to restore the sensor hub trigger;
5. Set the SHUB_PU_EN bit of the MASTER_CONFIG register to 0 in order to disable the I²C master pull-up;
6. Set the PASS_THROUGH_MODE bit of the MASTER_CONFIG register to 1 in order to enable the pass-through feature.

7.3.2 Pass-through feature disable

The procedure below has to be used in order to disable the pass-through:

1. Wait for the external microcontroller connected to the main I²C line to complete all running I²C operations. The pass-through must not be disabled in the middle of an I²C transaction;
2. Set the PASS_THROUGH_MODE bit of the MASTER_CONFIG register to 0.

At this point, the internal I²C master pull-up can be restored by setting the SHUB_PU_EN bit of the MASTER_CONFIG register to 1, and the auxiliary I²C master can be enabled by setting the MASTER_ON bit of the MASTER_CONFIG register to 1.

7.4 Sensor hub mode example

The configuration of the external sensors can be performed using the pass-through feature. This feature can be enabled by setting the PASS_THROUGH_MODE bit of the MASTER_CONFIG register to 1 and implements a direct access to the external sensor registers, allowing quick configuration.

The code provided below gives basic routines to configure a device in sensor hub mode. Three different snippets of code are provided here, in order to present how to easily perform a one-shot write or read operation, using slave 0, and how to set up slave 0 for continuously reading external sensor data.

The PASS_THROUGH_MODE bit is disabled in all these routines, in order to be as generic as possible.

One-shot read routine (using internal trigger) is described below. For simplicity, the routine uses the accelerometer configured at 104 Hz, without external pull-ups on the I²C auxiliary bus.

1.	Write 40h to FUNC_CFG_ACCESS	// Enable access to sensor hub registers
2.	Write EXT_SENS_ADDR | 01h to SLV0_ADD	// Configure external device address (EXT_SENS_ADDR) // Enable read operation (rw_0 = 1)
3.	Write REG to SLV0_SUBADD	// Configure address (REG) of the register to be read
4.	Write 01h to SLAVE0_CONFIG	// Read one byte, SHUB_ODR = 104 Hz

5.	Write 4Ch to MASTER_CONFIG	// WRITE_ONCE is mandatory for read
		// I²C master enabled, using slave 0 only
		// I²C pull-ups enabled on SDx and SCx

6.	Write 00h to FUNC_CFG_ACCESS	// Disable access to sensor hub registers
7.	Read OUTX_H_A register	// Clear accelerometer data-ready XLDA
8.	Poll STATUS_REG, until XLDA = 1	// Wait for sensor hub trigger
9.	Poll STATUS_MASTER_MAINPAGE, until SENS_HUB_ENDOP = 1	// Wait for sensor hub read transaction

10.	Write 40h to FUNC_CFG_ACCESS	// Enable access to sensor hub registers
11.	Write 08h to MASTER_CONFIG	// I²C master disable
12.	Wait 300 µs	
13.	Read SENSOR_HUB_1 register	// Retrieve the output of the read operation
14.	Write 00h to FUNC_CFG_ACCESS	// Disable access to sensor hub registers

The one-shot routine can be easily changed to setup the device for **continuous reading** of external sensor data:

1.	Write 40h to FUNC_CFG_ACCESS	// Enable access to sensor hub registers
2.	Write EXT_SENS_ADDR | 01h to SLV0_ADD	// Configure external device address (EXT_SENS_ADDR) // Enable read operation (rw_0 = 1)
3.	Write REG to SLV0_SUBADD	// Configure address (REG) of the register to be read
4.	Write 0xh to SLAVE0_CONFIG	// Read x bytes (up to six), SHUB_ODR = 104 Hz

5.	Write 4Ch to MASTER_CONFIG	// WRITE_ONCE is mandatory for read
		// I²C master enabled, using slave 0 only
		// I²C pull-ups enabled on SDx and SCx

6.	Write 00h to FUNC_CFG_ACCESS	// Disable access to sensor hub registers

After the execution of step 6, external sensor data are available to be read in sensor hub output registers.

The **One-shot write routine** (using internal trigger) is described below. For simplicity, the routine uses the accelerometer configured at 104 Hz, without external pull-ups on the I²C auxiliary bus.

1.	Write 40h to FUNC_CFG_ACCESS	// Enable access to sensor hub registers
2.	Write EXT_SENS_ADDR to SLV0_ADD	// Configure external device address (EXT_SENS_ADDR) // Enable write operation (rw_0 = 0)
3.	Write REG to SLV0_SUBADD	// Configure address (REG) of the register to be written
4.	Write 00h to SLAVE0_CONFIG	// SHUB_ODR = 104 Hz
5.	Write VAL to DATAWRITE_SLV0	// Configure value (VAL) to be written in REG

6.	Write 4Ch to MASTER_CONFIG	// WRITE_ONCE enabled for single write
		// I²C master enabled, using slave 0 only
		// I²C pull-ups enabled on SDx and SCx

7.	Poll STATUS_MASTER, until WR_ONCE_DONE = 1	// Wait for sensor hub write transaction
8.	Write 08h to MASTER_CONFIG	// I²C master disabled
9.	Wait 300 µs	
10.	Write 00h to FUNC_CFG_ACCESS	// Disable access to sensor hub registers

The following sequence configures the LIS2MDL external magnetometer sensor (refer to the datasheet for additional details) in continuous-conversion mode at 100 Hz (enabling temperature compensation, BDU and offset cancellation features) and reads the magnetometer output registers, saving their values in the SENSOR_HUB_1 to SENSOR_HUB_6 registers.

1.	Write 40h to CTRL1_XL	// Turn on the accelerometer (for trigger signal) at 104 Hz
2.	Perform **one-shot read** with	// Check LIS2MDL WHO_AM_I register
	SLV0_ADD = 3Dh	// LIS2MDL slave address is 3Ch and rw_0=1
	SLV0_SUBADD = 4Fh	// WHO_AM_I register address is 4Fh
3.	Perform **one-shot write** with	// Write LIS2MDL register CFG_REG_A (60h) = 8Ch
	SLV0_ADD = 3Ch	// LIS2MDL slave address is 3Ch and rw_0=0
	SLV0_SUBADD = 60h	// Enable temperature compensation
	DATAWRITE_SLV0 = 8Ch	// Enable magnetometer at 100 Hz ODR in continuous mode
4.	Perform **one-shot write** with	// Write LIS2MDL register CFG_REG_B (61h) = 02h
	SLV0_ADD = 3Ch	// LIS2MDL slave address is 3Ch and rw_0=0
	SLV0_SUBADD = 61h	// Enable magnetometer offset-cancellation
	DATAWRITE_SLV0 = 02h	
5.	Perform **one-shot write** with	// Write LIS2MDL register CFG_REG_B (62h) = 10h
	SLV0_ADD = 3Ch	// LIS2MDL slave address is 3Ch and rw_0=0
	SLV0_SUBADD = 62h	// Enable magnetometer BDU
	DATAWRITE_SLV0 = 10h	
6.	Setup **continuous read** with	// LIS2MDL slave address is 3Ch and rw_0=1
	SLV0_ADD = 3Dh	// Magnetometer output registers start from 68h
	SLV0_SUBADD = 68h	// Set up a continuous 6-byte read from I²C master interface
	SLAVE0_CONFIG = 06h	

8 First-in, first-out (FIFO) buffer

In order to limit intervention by the host processor and facilitate post-processing data for event recognition, the IIS2ICLX embeds a 3 kbyte first-in, first-out buffer (FIFO).

The FIFO can be configured to store the following data:

- accelerometer sensor data;
- timestamp data;
- temperature sensor data;
- external sensor (connected to sensor hub interface) data.

Saving the data in FIFO is based on FIFO words. A FIFO word is composed of :

- tag, 1 byte
- data, 6 bytes

Data can be retrieved from the FIFO through six dedicated registers, from address 79h to 7Eh: FIFO_DATA_OUT_X_L, FIFO_DATA_OUT_X_H, FIFO_DATA_OUT_Y_L, FIFO_DATA_OUT_Y_H, FIFO_DATA_OUT_Z_L, FIFO_DATA_OUT_Z_H.

The reconstruction of a FIFO stream is a simple task thanks to the FIFO_TAG field of FIFO_DATA_OUT_TAG register that allows recognizing the meaning of a word in FIFO. The applications have maximum flexibility in choosing the rate of batching for sensors with dedicated FIFO configurations.

Six different FIFO operating modes can be chosen through the FIFO_MODE[2:0] bits of the FIFO_CTRL4 register:

- Bypass mode;
- FIFO mode;
- Continuous mode;
- Continuous-to-FIFO mode;
- Bypass-to-Continuous mode;
- Bypass-to-FIFO mode.

To monitor the FIFO status (full, overrun, number of samples stored, etc.), two dedicated registers are available: FIFO_STATUS1 and FIFO_STATUS2.

Programmable FIFO threshold can be set in FIFO_CTRL1 and FIFO_CTRL2 using the WTM[8:0] bits.

FIFO full, FIFO threshold and FIFO overrun events can be enabled to generate dedicated interrupts on the two interrupt pins (INT1 and INT2) through the INT1_FIFO_FULL, INT1_FIFO_FTH and INT1_FIFO_OVR bits of the INT1_CTRL register, and through the INT2_ FIFO_FULL, INT2_FIFO_FTH and INT2_FIFO_OVR bits of the INT2_CTRL register.

8.1 FIFO description and batched sensors

FIFO is divided into 512 words of 7 bytes each. A FIFO word contains one byte with TAG information and 6 bytes of data: the overall FIFO buffer dimension is equal to 3584 bytes and can contain 3072 bytes of data. The TAG byte contains the information indicating which data is stored in the FIFO data field and other useful information.

FIFO is runtime configurable: a meta-information tag can be enabled in order to notify the user if batched sensor configurations have changed.

Batched sensors can be classified in three different categories:

1. Main sensor which is the accelerometer sensor;
2. Auxiliary sensors, which contain information of the status of the device:
   a. Timestamp sensor;
   b. Configuration-change sensor (CFG-Change);
   c. Temperature sensor;
3. Virtual sensors which are the external sensors read from the sensor hub interface.

Data can be retrieved from the FIFO through six dedicated registers: FIFO_DATA_OUT_X_L, FIFO_DATA_OUT_X_H, FIFO_DATA_OUT_Y_L, FIFO_DATA_OUT_Y_H, FIFO_DATA_OUT_Z_L, FIFO_DATA_OUT_Z_H.

A write to FIFO can be triggered by two different events:

- Accelerometer data-ready signal;
- Sensor hub data-ready.

8.2 FIFO registers

The FIFO buffer is managed by:

- Six control registers: FIFO_CTRL1, FIFO_CTRL2, FIFO_CTRL3, FIFO_CTRL4, COUNTER_BDR_REG1, COUNTER_BDR_REG2;
- Two status registers: FIFO_STATUS1 and FIFO_STATUS2;
- Seven output registers (tag + data): FIFO_DATA_OUT_TAG, FIFO_DATA_OUT_X_L, FIFO_DATA_OUT_X_H, FIFO_DATA_OUT_Y_L, FIFO_DATA_OUT_Y_H, FIFO_DATA_OUT_Z_L, FIFO_DATA_OUT_Z_H;
- Some additional bits to route FIFO events to the two interrupt lines: INT1_CNT_BDR, INT1_FIFO_FULL, INT1_FIFO_OVR, INT1_FIFO_TH bits of the INT1_CTRL register and INT2_CNT_BDR, INT2_FIFO_FULL, INT2_FIFO_OVR, INT2_FIFO_TH bits of the INT2_CTRL register;
- Some additional bits for other features include the BATCH_EXT_SENS_0_EN, BATCH_EXT_SENS_1_EN, BATCH_EXT_SENS_2_EN, BATCH_EXT_SENS_3_EN bits of the SLAVE0_CONFIG, SLAVE1_CONFIG, SLAVE2_CONFIG, SLAVE3_CONFIG sensor hub registers, which enable the batching in FIFO of the related external sensors.

8.2.1 FIFO_CTRL1

The FIFO_CTRL1 register contains the lower part of the 9-bit FIFO watermark threshold level. For the complete watermark threshold level configuration, consider also the WTM8 bit of the FIFO_CTRL2 register. 1 LSB value of the FIFO threshold level is referred to as a FIFO word (7 bytes).

The FIFO watermark flag (FIFO_WTM_IA bit in the FIFO_STATUS2 register) rises when the number of bytes stored in the FIFO is equal to or higher than the watermark threshold level.

In order to limit the FIFO depth to the watermark level, the STOP_ON_WTM bit must be set to 1 in the FIFO_CTRL2 register.

Table 31. FIFO_CTRL1 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0

8.2.2 FIFO_CTRL2

Table 32. **FIFO_CTRL2 register**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STOP_ ON_WTM	0	0	ODRCHG _EN	0	0	0	WTM8

The FIFO_CTRL2 register contains the upper part of the 9-bit FIFO watermark threshold level (WTM8 bit). For the complete watermark threshold level configuration, consider also the WTM[7:0] bits of the FIFO_CTRL1 register. The register contains the bit STOP_ON_WTM which allows limiting the FIFO depth to the watermark level.

Moreover, the FIFO_CTRL2 register contains the ODRCHG_EN bit which can be set to 1 in order to enable the CFG-Change auxiliary sensor to be batched in FIFO (described in the next sections).

8.2.3 FIFO_CTRL3

Table 33. **FIFO_CTRL3 register**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	BDR_XL_3	BDR_XL_2	BDR_XL_1	BDR_XL_0

The FIFO_CTRL3 register contains the field to select the writing frequency in FIFO for accelerometer sensor data. The selected batch data rate must be equal to or lower than the output data rate configured through the ODR_XL field of the CTRL1_XL register.

The following table indicate all the selectable batch data rates.

Table 34. **Accelerometer batch data rate**

BDR_XL[3:0]	Batch data rate [Hz]
0000	Not batched in FIFO
0001	12.5
0010	26
0011	52
0100	104
0101	208
0110	416
0111	833
1011	1.6

8.2.4 FIFO_CTRL4

The FIFO_CTRL4 register contains the fields to select the decimation factor for timestamp batching in FIFO and the batching data rate for the temperature sensor.

The timestamp writing rate is configured to the maximum rate between the accelerometer and sensor hub batch data rate divided by the decimation factor specified in the DEC_TS_BATCH_[1:0] field. The programmable decimation factors are indicated in the table below.

Table 35. Timestamp batch data rate

DEC_TS_BATCH[1:0]	Timestamp batch data rate [Hz]
00	Not batched in FIFO
01	max(BDR_XL[Hz], BDR_SHUB[Hz])
10	max(BDR_XL[Hz], BDR_SHUB[Hz]) / 8
11	max(BDR_XL[Hz], BDR_SHUB[Hz]) / 32

The temperature batch data rate is configurable through the ODR_T_BATCH_[1:0] field as shown in the table below.

Table 36. Temperature sensor batch data rate

ODR_T_BATCH[1:0]	Temperature batch data rate [Hz]
00	Not batched in FIFO
01	1.6
10	12.5
11	52

The FIFO_CTRL4 register also contains the FIFO operating modes bits. FIFO operating modes are described in Section 8.7 FIFO modes.

Table 37. FIFO_CTRL4 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DEC_TS_ BATCH_1	DEC_TS_ BATCH_0	ODR_T_ BATCH_1	ODR_T_ BATCH_0	0	FIFO_ MODE2	FIFO_ MODE1	FIFO_ MODE0

8.2.5 COUNTER_BDR_REG1

Since the FIFO might contain meta-information (i.e. CFG-Change sensor) and external sensor data, the FIFO provides a way to synchronize FIFO reading on the basis of the accelerometer actual number of samples stored in FIFO: the BDR counter.

The BDR counter can be configured through the COUNTER_BDR_REG1 and COUNTER_BDR_REG2 registers.

Table 38. COUNTER_BDR_REG1 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	RST_COUNTER_BDR	0	0	0	0	0	CNT_BDR_TH_8

RST_COUNTER_BDR can be asserted to reset the BDR counter: it is automatically reset to zero.

The user can select the threshold which generates the COUNTER_BDR_IA event in the FIFO_STATUS2 register. Once the internal BDR counter reaches the threshold, the COUNTER_BDR_IA bit is set to 1. The threshold is configurable through the CNT_BDR_TH_[8:0] bits. The upper part of the field is contained in register COUNTER_BDR_REG1. 1 LSB value of the CNT_BDR_TH threshold level is referred to as one accelerometer sample (X and Y axes data).

8.2.6 COUNTER_BDR_REG2

The COUNTER_BDR_REG2 register contains the lower part of the BDR-counter threshold.

Table 39. COUNTER_BDR_REG2 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CNT_BDR_TH_7	CNT_BDR_TH_6	CNT_BDR_TH_5	CNT_BDR_TH_4	CNT_BDR_TH_3	CNT_BDR_TH_2	CNT_BDR_TH_1	CNT_BDR_TH_0

8.2.7 FIFO_STATUS1

The FIFO_STATUS1 register, together with the FIFO_STATUS2 register, provides information about the number of samples stored in the FIFO. 1 LSB value of the DIFF_FIFO level is referred to as a FIFO word (7 bytes).

Table 40. FIFO_STATUS1 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DIFF_FIFO_7	DIFF_FIFO_6	DIFF_FIFO_5	DIFF_FIFO_4	DIFF_FIFO_3	DIFF_FIFO_2	DIFF_FIFO_1	DIFF_FIFO_0

8.2.8 FIFO_STATUS2

The FIFO_STATUS2 register, together with the FIFO_STATUS1 register, provides information about the number of samples stored in the FIFO and about the current status (watermark, overrun, full, BDR counter) of the FIFO buffer.

Table 41. FIFO_STATUS2 register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FIFO_ WTM_IA	FIFO_ OVR_IA	FIFO_ FULL_IA	COUNTER _BDR_IA	FIFO_OVR_ LATCHED	0	DIFF_ FIFO_9	DIFF_ FIFO_8

- FIFO_WTM_IA represents the watermark status. This bit goes high when the number of FIFO words (7 bytes each) already stored in the FIFO is equal to or higher than the watermark threshold level. The watermark status signal can be driven to the two interrupt pins by setting to 1 the INT1_FIFO_TH bit of the INT1_CTRL register or the INT2_FIFO_TH bit of the INT2_CTRL register.
- FIFO_OVR_IA goes high when the FIFO is completely filled and at least one sample has already been overwritten to store the new data. This signal can be driven to the two interrupt pins by setting to 1 the INT1_FIFO_OVR bit of the INT1_CTRL register or the INT2_FIFO_OVR bit of the INT2_CTRL register.
- FIFO_FULL_IA goes high when the next set of data that will be stored in FIFO will make the FIFO completely full (i.e. DIFF_FIFO_9 = 1) or generate a FIFO overrun. This signal can be driven to the two interrupt pins by setting to 1 the INT1_FIFO_FULL bit of the INT1_CTRL register or the INT2_FIFO_FULL bit of the INT2_CTRL register.
- COUNTER_BDR_IA represents the BDR-counter status. This bit goes high when the number of accelerometer batched samples reaches the BDR-counter threshold level configured through the CNT_BDR_TH_[8:0] bits of the COUNTER_BDR_REG1 and COUNTER_BDR_REG2 registers. The COUNTER_BDR_IA bit is automatically reset when the FIFO_STATUS2 register is read. The BDR counter status can be driven to the two interrupt pins by setting to 1 the INT1_CNT_BDR bit of the INT1_CTRL register or the INT2_CNT_BDR bit of the INT2_CTRL register.
- FIFO_OVR_LATCHED, as FIFO_OVR_IA, goes high when the FIFO is completely filled and at least one sample has already been overwritten to store the new data. The difference between the two flags is that FIFO_OVR_LATCHED is reset when the FIFO_STATUS2 register is read, whereas the FIFO_OVR_IA is reset when at least one FIFO word is read. This allows detecting a FIFO overrun condition during reading data from FIFO.
- DIFF_FIFO_[9:8] contains the upper part of the number of unread words stored in the FIFO. The lower part is represented by the DIFF_FIFO_[7:0] bits in FIFO_STATUS1. The value of the DIFF_FIFO_[9:0] field corresponds to the number of 7-byte words in the FIFO.

Register content is updated synchronously to the FIFO write and read operations.

Note: The BDU feature also acts on the FIFO_STATUS1 and FIFO_STATUS2 registers. When the BDU bit is set to 1, it is mandatory to read FIFO_STATUS1 first and then FIFO_STATUS2.

8.2.9 FIFO_DATA_OUT_TAG

By reading the FIFO_DATA_OUT_TAG register, it is possible to understand to which sensor the data of the current reading belongs and to check if data are consistent.

Table 42. FIFO_DATA_OUT_TAG register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TAG_ SENSOR_4	TAG_ SENSOR_3	TAG_ SENSOR_2	TAG_ SENSOR_1	TAG_ SENSOR_0	TAG_ CNT_1	TAG_ CNT_0	TAG_ PARITY

- TAG_SENSOR_[4:0] field identifies the sensors stored in the 6 data bytes (Table 43);
- TAG_CNT_[1:0] field identifies the FIFO time slot (described in next sections);
- TAG_PARITY bit recognizes if the content of the FIFO_DATA_OUT_TAG register is corrupted.

The table below contains all the possible values and associated type of sensor for the TAG_SENSOR_[4:0] field.

Table 43. TAG_SENSOR field and associated sensor

TAG_SENSOR_[4:0]	Sensor name	Sensor category	Description
0x02	Accelerometer	Main	Accelerometer data
0x03	Temperature	Auxiliary	Temperature data
0x04	Timestamp	Auxiliary	Timestamp data
0x05	CFG_Change	Auxiliary	Meta-information data
0x0E	Sensor Hub Slave 0	Virtual	Sensor hub data from slave 0
0x0F	Sensor Hub Slave 1	Virtual	Sensor hub data from slave 1
0x10	Sensor Hub Slave 2	Virtual	Sensor hub data from slave 2
0x11	Sensor Hub Slave 3	Virtual	Sensor hub data from slave 3
0x19	Sensor Hub Nack	Virtual	Sensor hub nack from slave 0/1/2/3

The TAG_PARITY bit can be used to check the content of the FIFO_DATA_OUT_TAG register. In order to do this, the user can implement the following routine:

1. Read the FIFO_DATA_OUT_TAG register;
2. Count the number of bits equal to 1;
3. If the number of bits equal to 1 is even, then the FIFO_DATA_OUT_TAG content is reliable, otherwise it is unreliable.

8.2.10 FIFO_DATA_OUT

Data can be retrieved from the FIFO through six dedicated registers, from address 79h to address 7Eh: FIFO_DATA_OUT_X_L, FIFO_DATA_OUT_X_H, FIFO_DATA_OUT_Y_L, FIFO_DATA_OUT_Y_H, FIFO_DATA_OUT_Z_L, FIFO_DATA_OUT_Z_H.

The FIFO output registers content depends on the sensor category and type, as described in the next section.

8.3 FIFO batched sensors

As previously described, batched sensors can be classified in three different categories:

1. Main sensor;
2. Auxiliary sensors;
3. Virtual sensors.

In this section, all the details about each category will be presented.

8.4 Main sensor

The main sensor is the accelerometer. The batch data rate can be configured through the BDR_XL_[3:0] field of the FIFO_CTRL3 register. The batch data rate must be equal to or lower than the sensor output data rate configured through the ODR_XL[3:0] field of the CTRL1_XL register.

The main sensor defines the FIFO time base. This means that each one of the other sensors can be associated to a time base slot defined by the main sensor. A batch event of the main sensor also increments the TAG counter (TAG_CNT field of FIFO_DATA_OUT_TAG register). This counter is composed of two bits and its value is continuously incremented (from 00b to 11b) to identify different time slots.

The FIFO word format of the main sensor is presented in the table below, representing the device addresses from 78h to 7Eh.

Table 44. Main sensor output data format in FIFO

TAG	X_L	X_H	Y_L	Y_H	0	0

8.5 Auxiliary sensors

Auxiliary sensors are considered as service sensors for the main sensor. Auxiliary sensors include the:

- Temperature sensor (ODR_T_BATCH_[1:0] bits of the FIFO_CTRL4 register must be configured properly);
- Timestamp sensor: it stores the timestamp corresponding to a FIFO time slot (TIMESTAMP_EN bit of the CTRL10_C register must be set to 1 and the DEC_TS_BATCH_[1:0] bits of the FIFO_CTRL4 register must be configured properly);
- CFG-Change sensor: it identifies a change in some configuration of the device (ODRCHG_EN bit of the FIFO_CTRL2 register must be set to 1).

Auxiliary sensors cannot trigger a write in FIFO. Their registers are written when the first main sensor or the external sensor event occurs (even if they are configured at a higher batching data rate).

The temperature output data format in FIFO is presented in the following table.

Table 45. Temperature output data format in FIFO

Data	FIFO_DATA_OUT registers
TEMPERATURE[7:0]	FIFO_DATA_OUT_X_L
TEMPERATURE[15:8]	FIFO_DATA_OUT_X_H
0	FIFO_DATA_OUT_Y_L
0	FIFO_DATA_OUT_Y_H
0	FIFO_DATA_OUT_Z_L
0	FIFO_DATA_OUT_Z_H

The timestamp output data format in FIFO is presented in the following table.

Table 46. Timestamp output data format in FIFO

Data	FIFO_DATA_OUT registers
TIMESTAMP[7:0]	FIFO_DATA_OUT_X_L
TIMESTAMP[15:8]	FIFO_DATA_OUT_X_H
TIMESTAMP[23:16]	FIFO_DATA_OUT_Y_L
TIMESTAMP[31:24]	FIFO_DATA_OUT_Y_H
BDR_SHUB	FIFO_DATA_OUT_Z_L[3:0]
0	FIFO_DATA_OUT_Z_L[7:4]
BDR_XL	FIFO_DATA_OUT_Z_H[3:0]
0	FIFO_DATA_OUT_Z_H[7:4]

As shown in Table 46, timestamp data contain also some meta-information which can be used to detect a BDR change if the CFG-Change sensor is not batched in FIFO: the batching data rate of both the main sensor and the sensor hub. BDR_SHUB cannot be configured though a dedicated register. It is the result of the configured sensor hub ODR through the SHUB_ODR_[1:0] bits of the SLAVE0_CONFIG sensor hub register and the effective trigger sensor output data rate (the accelerometer output data rate if the internal trigger is used). For the complete description of BDR_SHUB, refer to to the next section about virtual sensors.

CFG-Change identifies a runtime change in the output data rate, the batching data rate or other configurations of the main or virtual sensors. When a supported runtime change is applied, this sensor is written at the first new main sensor or virtual sensor event followed by a timestamp sensor (also if the timestamp sensor is not batched).

This sensor can be used to correlate data from the sensors to the device timestamp without storing the timestamp each time. It could be used also to notify the user to discard data due to embedded filters settling or to other configuration changes (i.e switching mode, output data rate, …).

CFG-Change output data format in FIFO is presented in the following table.

Table 47. CFG-change output data format in FIFO

Data	FIFO_DATA_OUT registers
0	FIFO_DATA_OUT_X_H[7:0]
LPF2_XL_EN	FIFO_DATA_OUT_Y_L[0]
HPCF_XL_[2:0]	FIFO_DATA_OUT_Y_L[3:1]
0	FIFO_DATA_OUT_Y_L[4]
0	FIFO_DATA_OUT_Y_L[5]
FS[1:0]_XL	FIFO_DATA_OUT_Y_L[7:6]
BDR_SHUB	FIFO_DATA_OUT_Y_H[3:0]
0	FIFO_DATA_OUT_Y_H[7:5]
ODR_XL	FIFO_DATA_OUT_Z_L[3:0]
0	FIFO_DATA_OUT_Z_L[7:4]
BDR_XL	FIFO_DATA_OUT_Z_H[3:0]
0	FIFO_DATA_OUT_Z_H[7:4]

8.6 Virtual sensors

Virtual sensors are the external sensors read from the sensor hub interface.

8.6.1 External sensors and NACK sensor

Data of up to four external sensors read from the sensor hub (for a maximum of 18 bytes) can be stored in FIFO.

They are continuous virtual sensors with the batching data rate (BDR_SHUB) corresponding to the current value of the SHUB_ODR_[1:0] field in the SLAVE0_CONFIG register, if an internal trigger is used (sensor hub read triggered by the accelerometer data-ready signal). This value is limited by the effective trigger sensor output data rate. If external sensors are not batched or an external trigger is used, BDR_SHUB is set to 0.

The following table shows the possible values of the BDR_SHUB field.

Table 48. BDR_SHUB

BDR_SHUB	BDR [Hz]
0000	Not batched or external trigger used
0001	12.5
0010	26
0011	52
0100	104

As the main sensor, external sensors define the FIFO time base and they can trigger the writing of auxiliary sensors in FIFO (only if they are batched and an external trigger is not used).

It is possible to enable selectively the batching of the different external sensors using the BATCH_EXT_SENS_0_EN, BATCH_EXT_SENS_1_EN, BATCH_EXT_SENS_2_EN, BATCH_EXT_SENS_3_EN bits of the SLAVE0_CONFIG, SLAVE1_CONFIG, SLAVE2_CONFIG, SLAVE3_CONFIG sensor hub registers.

Each external sensor has a dedicated TAG value and 6 bytes reserved for data. External sensors are written in FIFO in the same order of the sensor hub output registers and if the number of bytes read from an external sensor is less than 6 bytes, then free bytes are filled with zeros.

If the communication with one external sensor batched in FIFO fails, the sensor hub writes a NACK sensor instead of the corresponding sensor data in FIFO. A NACK sensor contains the index (numbered from 0 to 3) of the failing slave and has the following output data format.

Table 49. Nack sensor output data format in FIFO

Data	FIFO_DATA_OUT registers
Failing slave index	FIFO_DATA_OUT_X_L[1:0]
0	FIFO_DATA_OUT_X_L[7:2]
0	FIFO_DATA_OUT_X_H
0	FIFO_DATA_OUT_Y_L
0	FIFO_DATA_OUT_Y_H
0	FIFO_DATA_OUT_Z_L
0	FIFO_DATA_OUT_Z_H

8.7 FIFO modes

The IIS2ICLX FIFO buffer can be configured to operate in six different modes, selectable through the FIFO_MODE_[2:0] field of the FIFO_CTRL4 register. The available configurations ensure a high level of flexibility and extend the number of functions usable in application development.

Bypass, FIFO, Continuous, Continuous-to-FIFO, Bypass-to-Continuous, and Bypass-to-FIFO modes are described in the following paragraphs.

8.7.1 Bypass mode

When Bypass mode is enabled, the FIFO is not used, the buffer content is cleared, and it remains empty until another mode is selected. Bypass mode is selected when the FIFO_MODE_[2:0] bits are set to 000b. Bypass mode must be used in order to stop and reset the FIFO buffer when a different mode is intended to be used. Note that by placing the FIFO buffer into Bypass mode, the whole buffer content is cleared.

8.7.2 FIFO mode

In FIFO mode, the buffer continues filling until it becomes full. Then it stops collecting data and the FIFO content remains unchanged until a different mode is selected.

Follow these steps for FIFO mode configuration:

1. Enable the sensor data to be stored in FIFO and relative batching data rate (if configurable);
2. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 001b to enable FIFO mode.

When this mode is selected, the FIFO starts collecting data. The FIFO_STATUS1 and FIFO_STATUS2 registers are updated according to the number of samples stored.

When the FIFO is full, the DIFF_FIFO_9 bit of the FIFO_STATUS2 register is set to 1 and no more data are stored in the FIFO buffer. Data can be retrieved by reading all the FIFO_DATA_OUT (from 78h to 7Eh) registers for the number of times specified by the DIFF_FIFO_[9:0] bits of the FIFO_STATUS1 and FIFO_STATUS2 registers.

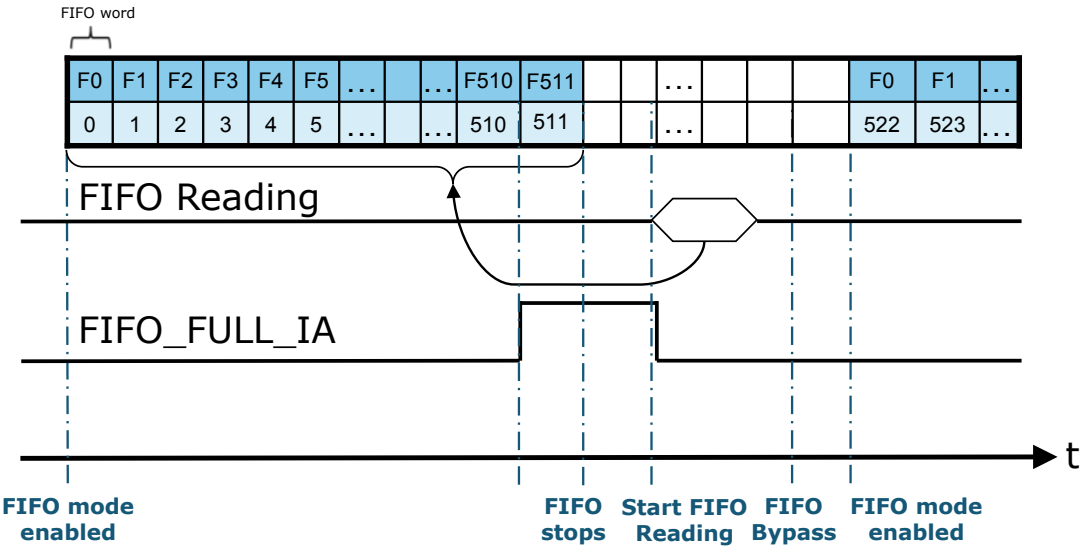
Using the FIFO_WTM_IA bit of the FIFO_STATUS2 register, data can also be retrieved when a threshold level (WTM[8:0] in FIFO_CTRL1 and FIFO_CTRL2 registers) is reached if the application requires a lower number of samples in the FIFO.

If the STOP_ON_WTM bit of the FIFO_CTRL2 register is set to 1, the FIFO size is limited to the value of the WTM[8:0] bits in the FIFO_CTRL1 and FIFO_CTRL2 registers. In this case, the FIFO_FULL_IA bit of the FIFO_STATUS2 register is set high when the number of samples in FIFO will reach or exceed the WTM[8:0] value on the next FIFO write operation.

Communication speed is not very important in FIFO mode because the data collection is stopped and there is no risk of overwriting data already acquired. Before restarting the FIFO mode, it is necessary to set to Bypass mode first in order to completely clear the FIFO content.

Figure 19. FIFO mode (STOP_ON_WTM = 0) shows an example of FIFO mode usage; the data from just one sensor are stored in the FIFO. In these conditions, the number of samples that can be stored in the FIFO buffer is 512. The FIFO_FULL_IA bit of the FIFO_STATUS2 register goes high just after the level labeled as 510 to notify that the FIFO buffer will be completely filled at the next FIFO write operation. After the FIFO is full (FIFO_DIFF_9 = 1), the data collection stops.

Figure 19. FIFO mode (STOP_ON_WTM = 0)



8.7.3 Continuous mode

In Continuous mode, the FIFO continues filling. When the buffer is full, the FIFO index restarts from the beginning, and older data are replaced by the new data. The oldest values continue to be overwritten until a read operation frees FIFO slots. The host processor reading speed is important in order to free slots faster than new data is made available. To stop this configuration, Bypass mode must be selected.

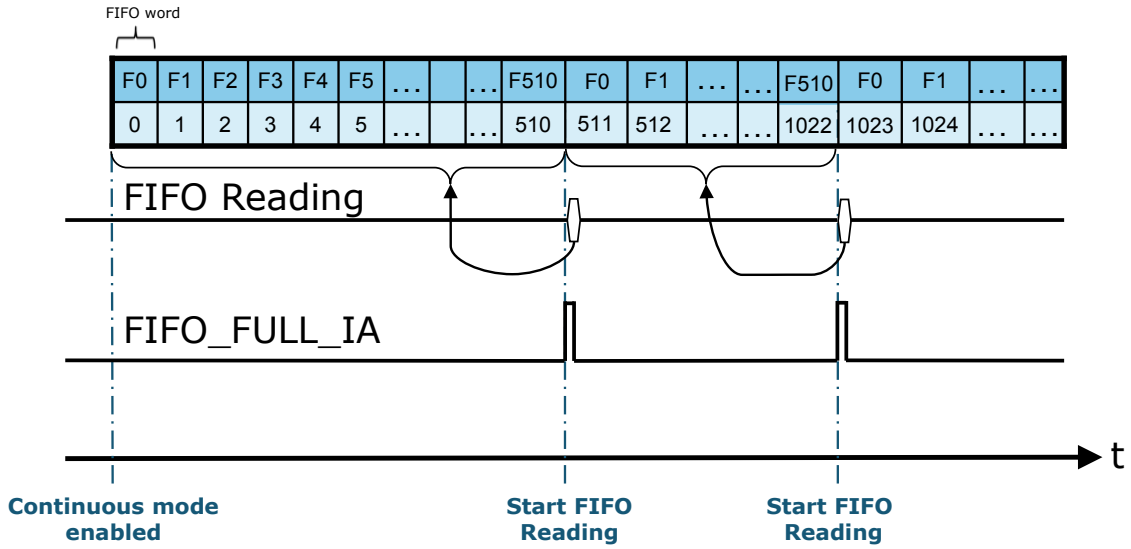
Follow these steps for Continuous mode configuration (if the accelerometer data-ready is used as the FIFO trigger):

1. Enable the sensor data to be stored in FIFO with the corresponding batching data rate (if configurable);
2. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 110b to enable FIFO mode.

When this mode is selected, the FIFO collects data continuously. The FIFO_STATUS1 and FIFO_STATUS2 registers are updated according to the number of samples stored. When the next FIFO write operation will make the FIFO completely full or generate a FIFO overrun, the FIFO_FULL_IA bit of the FIFO_STATUS2 register goes to 1. The FIFO_OVR_ IA and FIFO_OVR_LATCHED bits in the FIFO_STATUS2 register indicates when at least one FIFO word has been overwritten to store the new data. Data can be retrieved after the FIFO_FULL_IA event by reading the FIFO_DATA_OUT (from 78h to 7Eh) registers for the number of times specified by the DIFF_FIFO_[9:0] bits in the FIFO_STATUS1 and FIFO_STATUS2 registers. Using the FIFO_WTM_IA bit of the FIFO_STATUS2 register, data can also be retrieved when a threshold level (WTM[8:0] in the FIFO_CTRL1 and FIFO_CTRL2 registers) is reached. If the STOP_ON_WTM bit of the FIFO_CTRL2 register is set to 1, the FIFO size is limited to the value of the WTM[8:0] bits in the FIFO_CTRL1 and FIFO_CTRL2 registers. In this case, the FIFO_FULL_IA bit of the FIFO_STATUS2 register goes high when the number of samples in FIFO will reach or overcome the WTM[8:0] value on the next FIFO write operation.

Figure 20. Continuous mode shows an example of the Continuous mode usage. In the example, data from just one sensor are stored in the FIFO and the FIFO samples are read on the FIFO_FULL_IA event and faster than 1 * ODR so that no data is lost. In these conditions, the number of samples stored is 511.

Figure 20. Continuous mode



8.7.4 Continuous-to-FIFO mode

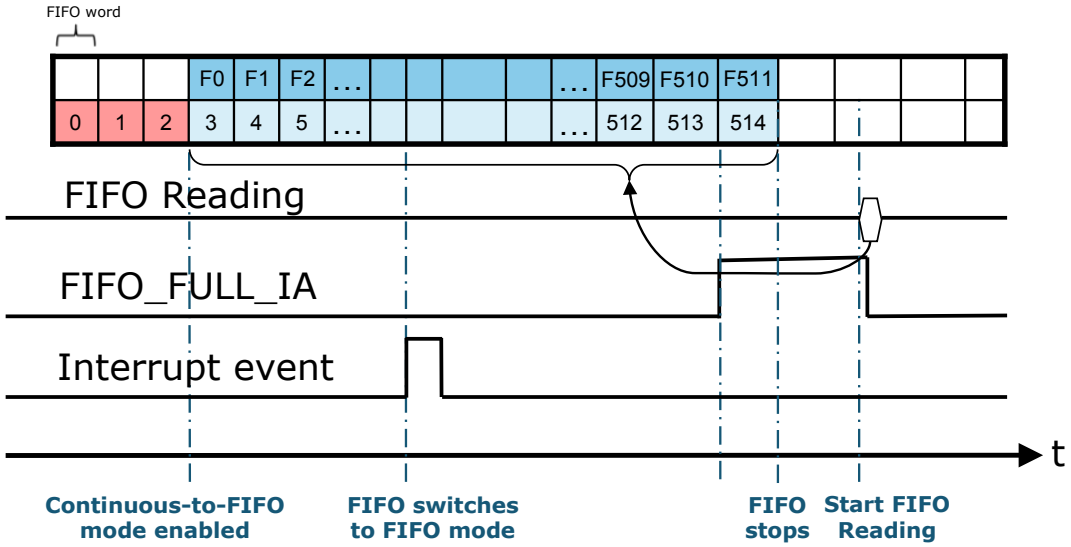
This mode is a combination of the Continuous and FIFO modes previously described. In Continuous-to-FIFO mode, the FIFO buffer starts operating in Continuous mode and switches to FIFO mode when an event condition occurs.

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1;
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1;
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1;

Continuous-to-FIFO mode is sensitive to the edge of the interrupt signal. At the first interrupt event, FIFO changes from Continuous mode to FIFO mode and maintains it until Bypass mode is set.

Figure 21. Continuous-to-FIFO mode



Follow these steps for Continuous-to-FIFO mode configuration (if the accelerometer data-ready is used as the FIFO trigger):

1. Configure one of the events as previously described;
2. Enable the sensor data to be stored in FIFO with the corresponding batching data rate (if configurable);
3. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 011b to enable FIFO Continuous-to-FIFO mode.

In Continuous-to-FIFO mode the FIFO buffer continues filling. When the FIFO will be full or overrun at the next FIFO write operation, the FIFO_FULL_IA bit goes high.

If the STOP_ON_WTM bit of the FIFO_CTRL2 register is set to 1, the FIFO size is limited to the value of the WTM[8:0] bits in the FIFO_CTRL1 and FIFO_CTRL2 registers. In this case, the FIFO_FULL_IA bit of the FIFO_STATUS2 register goes high when the number of samples in FIFO will reach or exceed the WTM[8:0] value on the next FIFO write operation.

When the trigger event occurs, two different cases can be observed:

1. If the FIFO buffer is already full, it stops collecting data at the first sample after the event trigger. The FIFO content is composed of the samples collected before the event.
2. If FIFO buffer is not full yet, it continues filling until it becomes full and then it stops collecting data.

Continuous-to-FIFO can be used in order to analyze the history of the samples which have generated an interrupt. The standard operation is to read the FIFO content when the FIFO mode is triggered and the FIFO buffer is full and stopped.

8.7.5 Bypass-to-Continuous mode

This mode is a combination of the Bypass and Continuous modes previously described. In Bypass-to-Continuous mode, the FIFO buffer starts operating in Bypass mode and switches to Continuous mode when an event condition occurs.

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1;
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1;
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1;

Bypass-to-Continuous mode is sensitive to the edge of the interrupt signal: at the first interrupt event, FIFO changes from Bypass mode to Continuous mode and maintains it until Bypass mode is set.

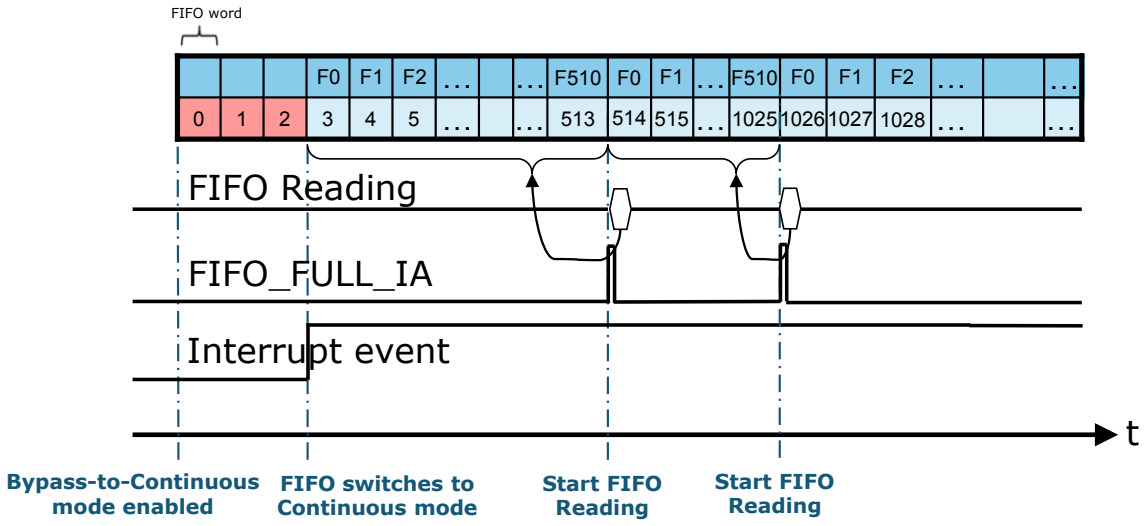
Follow these steps for Bypass-to-Continuous mode configuration (if the accelerometer data-ready is used as the FIFO trigger):

1. Configure one of the events as previously described;
2. Enable the sensor data to be stored in FIFO with the corresponding batching data rate (if configurable);
3. Set the FIFO_MODE[2:0] bits in the FIFO_CTRL4 register to 100b to enable FIFO Bypass-to-Continuous mode.

Once the trigger condition appears and the buffer switches to Continuous mode, the FIFO buffer continues filling. When the next stored set of data will make the FIFO full or overrun, the FIFO_FULL_IA bit is set high.

Bypass-to-Continuous can be used in order to start the acquisition when the configured interrupt is generated.

Figure 22. Bypass-to-Continuous mode



8.7.6 Bypass-to-FIFO mode

This mode is a combination of the Bypass and FIFO modes previously described. In Bypass-to-FIFO mode, the FIFO buffer starts operating in Bypass mode and switches to FIFO mode when an event condition occurs.

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1;
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1;
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1;

Bypass-to-FIFO mode is sensitive to the edge of the interrupt signal. At the first interrupt event, FIFO changes from Bypass mode to FIFO mode and maintains it until Bypass mode is set.

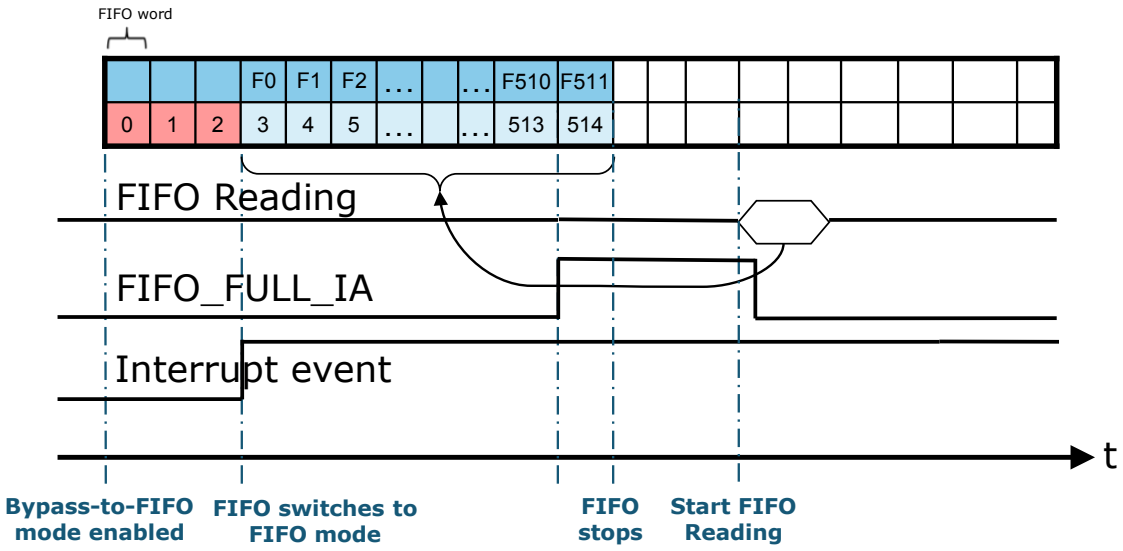
Follow these steps for Bypass-to-FIFO mode configuration (if the accelerometer data-ready is used as the FIFO trigger):

1. Configure one of the events as previously described;
2. Enable the sensor data to be stored in FIFO with the corresponding batching data rate (if configurable);
3. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 111b to enable FIFO Bypass-to-FIFO mode.

Once the trigger condition appears and the buffer switches to FIFO mode, the FIFO buffer starts filling. When the next stored set of data will make the FIFO full or overrun, the FIFO_FULL_IA bit is set high and the FIFO stops.

Bypass-to-FIFO can be used in order to analyze the history of the samples which have generated an interrupt.

Figure 23. Bypass-to-FIFO mode



8.8 Retrieving data from the FIFO

When FIFO is enabled and the mode is different from Bypass, reading the FIFO output registers return the oldest FIFO sample set. Whenever these registers are read, their content is moved to the SPI/I²C output buffer.

FIFO slots are ideally shifted up one level in order to release room for a new sample, and the FIFO output registers load the current oldest value stored in the FIFO buffer.

The recommended way to retrieve data from the FIFO is the following:

1. Read the FIFO_STATUS1 and FIFO_STATUS2 registers to check how many words are stored in the FIFO. This information is contained in the DIFF_FIFO_[9:0] bits.
2. For each word in FIFO, read the FIFO word (tag and output data) and interpret it on the basis of the FIFO tag.
3. Go to step 1.

The entire FIFO content is retrieved by performing a certain number of read operations from the FIFO output registers until the buffer becomes empty (DIFF_FIFO_[9:0] bits of the FIFO_STATUS1 and FIFO_STATUS2 register are equal to 0).

It is recommended to avoid reading from FIFO when it is empty.

FIFO output data must be read with multiple of 7 bytes reads starting from the FIFO_DATA_OUT_TAG register. The rounding function from address FIFO_DATA_OUT_Z_H to FIFO_DATA_OUT_TAG is done automatically in the device, in order to allow reading many words with a unique multiple read operation.

8.9 FIFO watermark threshold

The FIFO threshold is a functionality of the IIS2ICLX FIFO which can be used to check when the number of samples in the FIFO reaches a defined watermark threshold level.

The bits WTM[8:0] in the FIFO_CTRL1 and FIFO_CTRL2 registers contain the watermark threshold level. The resolution of the WTM[8:0] field is 7 bytes, corresponding to a complete FIFO word. So, the user can select the desired level in a range between 0 and 511.

The bit FIFO_WTM_IA in the FIFO_STATUS2 register represents the watermark status. This bit is set high if the number of words in the FIFO reaches or exceeds the watermark level. FIFO size can be limited to the threshold level by setting the STOP_ON_WTM bit in the FIFO_CTRL2 register to 1.

Figure 24. FIFO threshold (STOP_ON_WTM = 0)

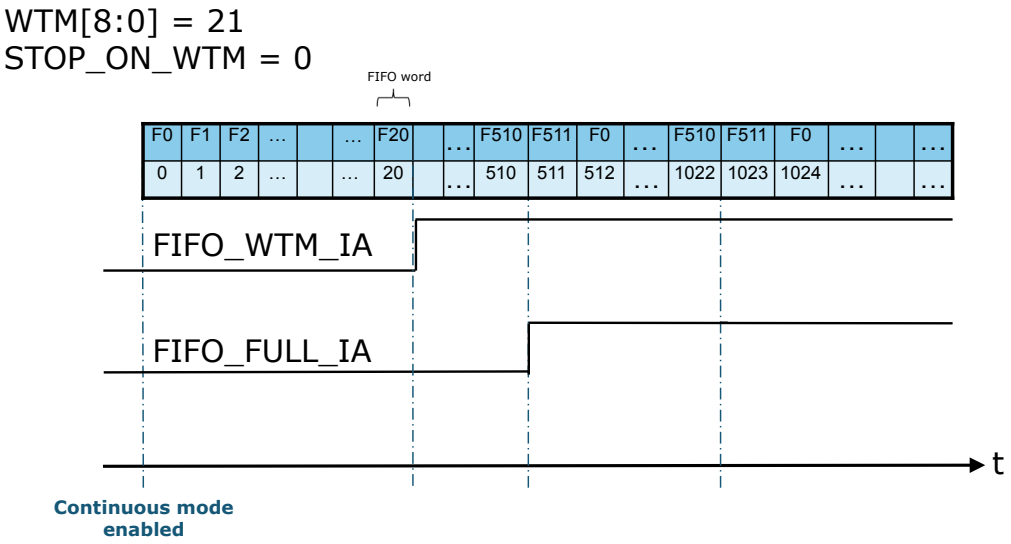


Figure 24. FIFO threshold (STOP_ON_WTM = 0) shows an example of FIFO threshold level usage when just accelerometer data are stored. The STOP_ON_WTM bit set to 0 in the FIFO_CTRL2 register. The threshold level is set to 21 through the WTM[8:0] bits. The FIFO_WTM_IA bit of the FIFO_STATUS2 register rises after the 21st level has been reached (21 words in the FIFO). Since the STOP_ON_WTM bit is set to 0, the FIFO will not stop at the 21st set of data, but will keep storing data until the FIFO_FULL_IA flag is set high.

Figure 25. **FIFO threshold (STOP_ON_WTM = 1) in FIFO mode**

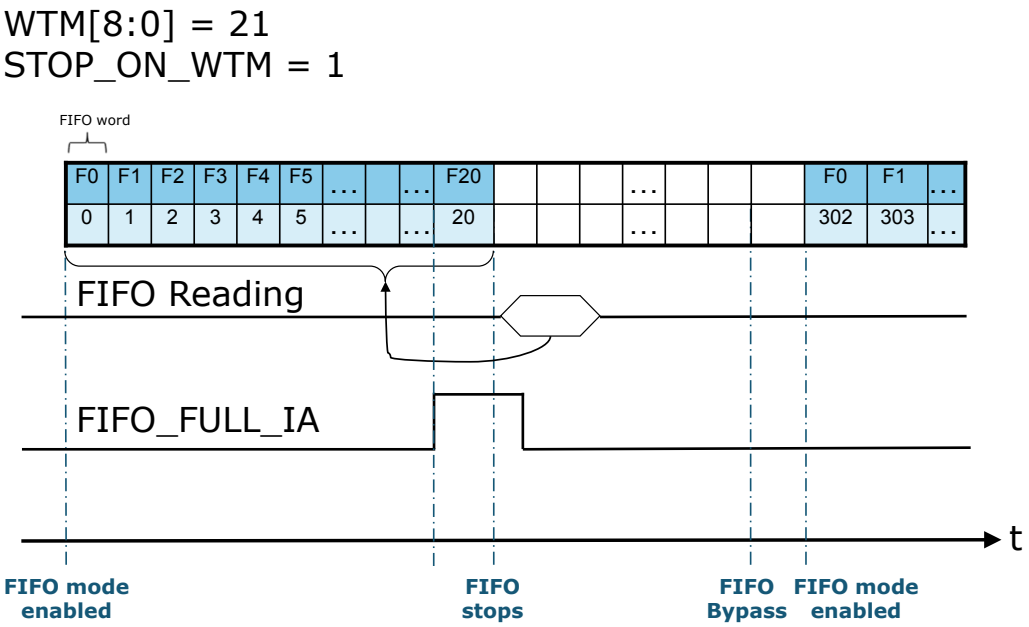


Figure 25. FIFO threshold (STOP_ON_WTM = 1) in FIFO mode shows an example of FIFO threshold level usage in FIFO mode with the STOP_ON_WTM bit set to 1 in the FIFO_CTRL2 register. Just accelerometer data are stored in this example. The threshold level is set to 21 through the WTM[8:0] bits and defines the current FIFO size. In FIFO mode, data are stored in the FIFO buffer until the FIFO is full. The FIFO_FULL_IA bit of the FIFO_STATUS2 register rises when the next data stored in the FIFO will generate the FIFO full or overrun condition. The FIFO_WTM_IA bit of the FIFO_STATUS2 register goes high when the FIFO is full.

Figure 26. **FIFO threshold (STOP_ON_WTM = 1) in Continuous mode**

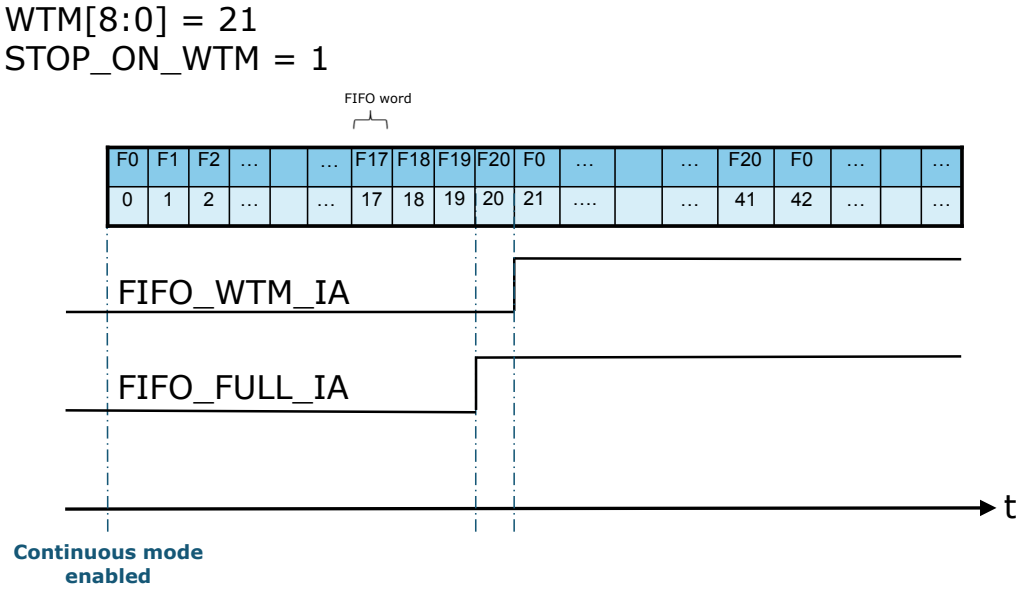


Figure 26. FIFO threshold (STOP_ON_WTM = 1) in Continuous mode shows an example of FIFO threshold level usage in Continuous mode with the STOP_ON_WTM bit set to 1 in the FIFO_CTRL2 register. Just accelerometer data are stored in this example. The threshold level is set to 21 through the WTM[8:0] bits. The FIFO_FULL_IA bit of the FIFO_STATUS2 register rises when the next data stored in the FIFO will make the FIFO full. The FIFO_WTM_IA bit of the FIFO_STATUS2 goes high when the FIFO is full. If data are not retrieved from FIFO, new data (labeled as sample 21) will override the older data stored in FIFO (labeled as sample F0).

8.10 Timestamp correlation

It is possible to reconstruct the timestamp of FIFO stream with three different approaches:

1. Basic, using only timestamp sensor information;
2. Memory-saving, based on the TAG_CNT field in FIFO_DATA_OUT_TAG
3. Hybrid, based on combined usage of the TAG_CNT field and decimated timestamp sensor

The basic approach guarantees the highest precision in timestamp reconstruction but wastes a lot of memory space available in FIFO. The timestamp sensor is written in FIFO at each time slot. If the overrun condition occurs, the correct procedure to retrieve the data from FIFO is to discard each data read before a new timestamp sensor.

The memory-saving approach uses only the TAG_CNT information and, when the TAG_CNT value increases, the timestamp stored at the software layer should be updated as follows:

$$timestamp = timestamp\left(i-1\right) + \frac{1}{\max(BDR\_XL, BDR\_SHUB)}$$

The memory-saving approach allows the user to maximize the data stored in FIFO. With this method all the timestamp correlation is forwarded to the application processor.

This approach is not recommended when the overrun condition can occur.

The hybrid approach is a trade-off and a combination of the two previous solutions. The timestamp is configured to be written in FIFO with decimation. When the TAG_CNT value increases, the timestamp stored at the software layer should be updated as in the memory-saving approach, while when the timestamp sensor is read, the timestamp stored at the software layer should be realigned with the correct value from the sensor.

9 Temperature sensor

The device is provided with an internal temperature sensor that is suitable for ambient temperature measurement.

If the accelerometer sensor is in Power-Down mode, the temperature sensor is off.

When the accelerometer is enabled, the output data rate of the temperature sensor is 52 Hz.

For the temperature sensor, the data-ready signal is represented by the TDA bit of the STATUS_REG register. The signal can be driven to the INT2 pin by setting the INT2_DRDY_TEMP bit of the INT2_CTRL register to 1.

The temperature data is given by the concatenation of the OUT_TEMP_H and OUT_TEMP_L registers and it is represented as a number of 16 bits in two's complement format with a sensitivity of 256 LSB/°C. The output zero level corresponds to 25 °C. Absolute temperature accuracy can be improved, reducing the effect of temperature offset, by performing OPC (one-point calibration) at room temperature (25 °C).

Temperature sensor data can also be stored in FIFO with a configurable batch data rate (see Section 8 First-in, first-out (FIFO) buffer for details).

9.1 Example of temperature data calculation

The following table provides a few basic examples of the data that is read from the temperature data registers at different ambient temperature values. The values listed in this table are given under the hypothesis of perfect device calibration (i.e. no offset, no gain error,....).

Table 50. Content of output data registers vs. temperature

Temperature values	Register address	
	OUT_TEMP_H (21h)	OUT_TEMP_L (20h)
0 °C	E7h	00h
25 °C	00h	00h
50 °C	19h	00h

10 Self-test

The embedded self-test functions allow checking the device functionality without moving it.

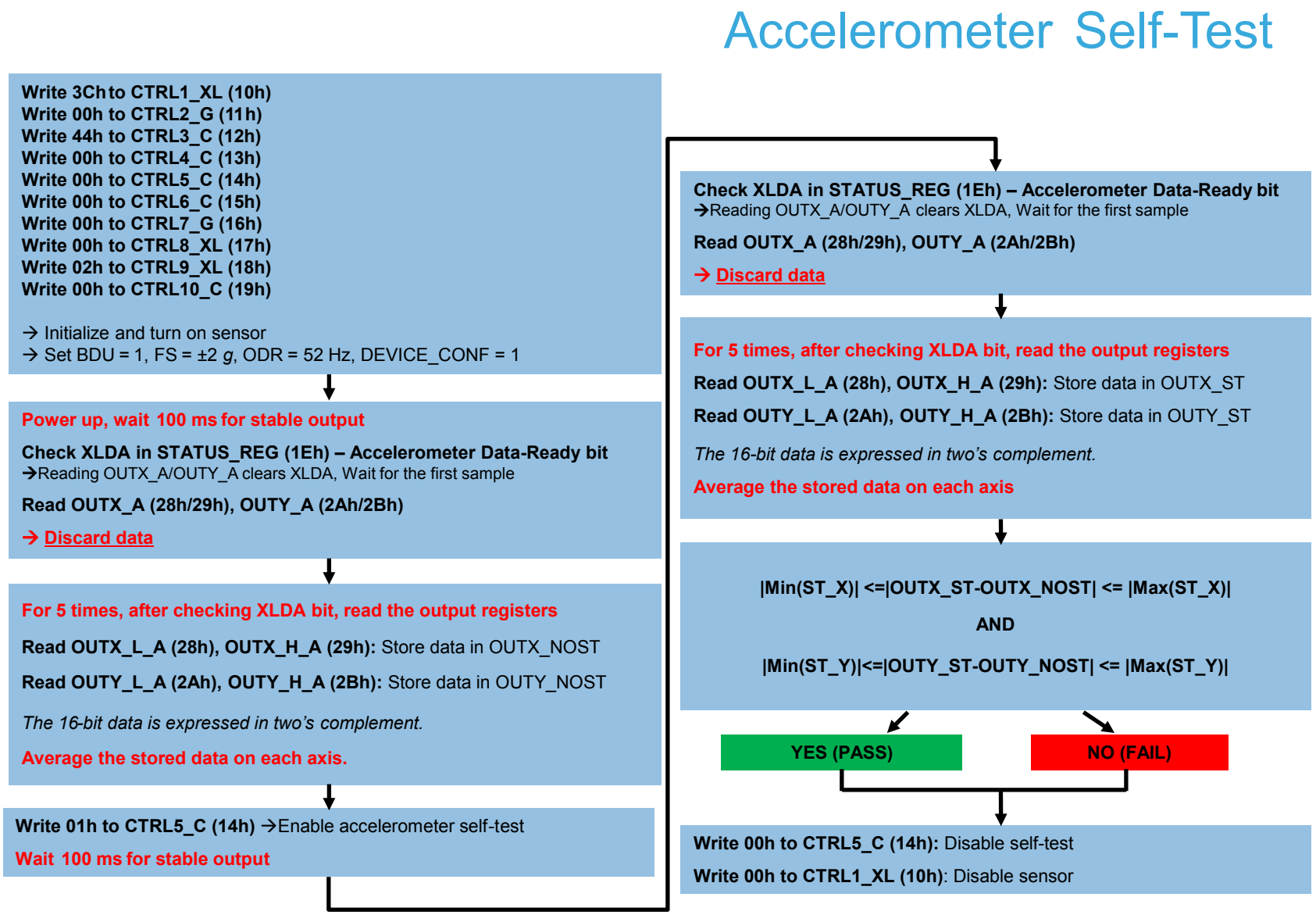
When the accelerometer self-test is enabled, an actuation force is applied to the sensor, simulating a definite input acceleration. In this case, the sensor outputs exhibit a change in their DC levels which are related to the selected full scale through the sensitivity value.

The accelerometer self-test is off when the ST[1:0]_XL bits of the CTRL5_C register are programmed to 00b; it is enabled when the ST[1:0]_XL bits are set to 01b (positive sign self-test) or 10b (negative sign self-test).

When the accelerometer self-test is activated, the sensor output level is given by the algebraic sum of the signals produced by the acceleration acting on the sensor and by the electrostatic test-force.

The complete accelerometer self-test procedure is indicated in Figure 27. Accelerometer self-test procedure.

Figure 27. **Accelerometer self-test procedure**



Revision history

Table 51. **Document revision history**

Date	Version	Changes
04-Aug-2020	1	Initial release
19-Oct-2020	2	Updated Table 3. Embedded functions registers Updated Table 6. Accelerometer ODR and power mode selection

Contents

List of tables

List of figures