LIS2DUXS12: ultralow-power 3-axis smart accelerometer with antialiasing filter, artificial intelligence, advanced digital features, and Qvar

Introduction

This document provides usage information and application hints related to ST's LIS2DUXS12 motion sensor.

The LIS2DUXS12 is a 3-axis digital accelerometer system-in-package with a digital I²C/SPI/MIPI I3C® serial interface standard output, performing at 10.8 µA in high-performance mode, at 6.2 µA in low-power mode (at 50 Hz), and at 2.7 µA in ultralow-power mode (at 1.6 Hz). The accelerometer features smart sleep-to-wake-up (activity) and return-to-sleep (inactivity) functions that allow advanced power saving. The embedded self-test capability allows the user to check that the sensor works in the final application.

The device has a dynamic user-selectable full-scale acceleration range of ±2/±4/±8/±16 *g* and can measure accelerations with output data rates from 1.6 Hz to 800 Hz. The LIS2DUXS12 can be configured to generate interrupt signals by using hardware recognition of free-fall events, 6D orientation, single/double/triple-tap sensing, activity or inactivity, and wake-up events.

The LIS2DUXS12 is compatible with the requirements of the leading OSs, offering real, virtual, and batch-mode sensors. It has been designed to implement in hardware significant motion, relative tilt, pedometer functions, timestamp, and provides an incredible level of customization: up to eight embedded finite state machines (FSM) can be programmed independently for motion detection or gesture recognition such as glance, absolute wrist tilt, shake, double-shake, or pick-up.

The LIS2DUXS12 also embeds machine learning core (MLC) logic that allows identifying if a data pattern matches a user-defined set of classes. A typical example of an application could be activity detection like running, walking, driving, and so forth.

The LIS2DUXS12 has an integrated 128-level first-in, first-out (FIFO) buffer allowing the user to store acceleration and temperature data in order to limit intervention by the host processor.

The device embeds an analog hub sensing functionality that is able to connect an analog input and convert it to a digital signal for embedded processing. In addition, an embedded Qvar (electric charge variation detection) channel can be used for motion detection, touch detection, and user interface (UI) applications.

The LIS2DUXS12 is available in a small thin plastic, land grid array (LGA) package, and it is guaranteed to operate over an extended temperature range from -40°C to +85°C.

The ultrasmall size and weight of the SMD package make it an ideal choice for handheld portable applications such as game controllers, IoT connected devices, and wearables or any other application where reduced package size and weight are required.

1 Pin description

Figure 1. Pin connections

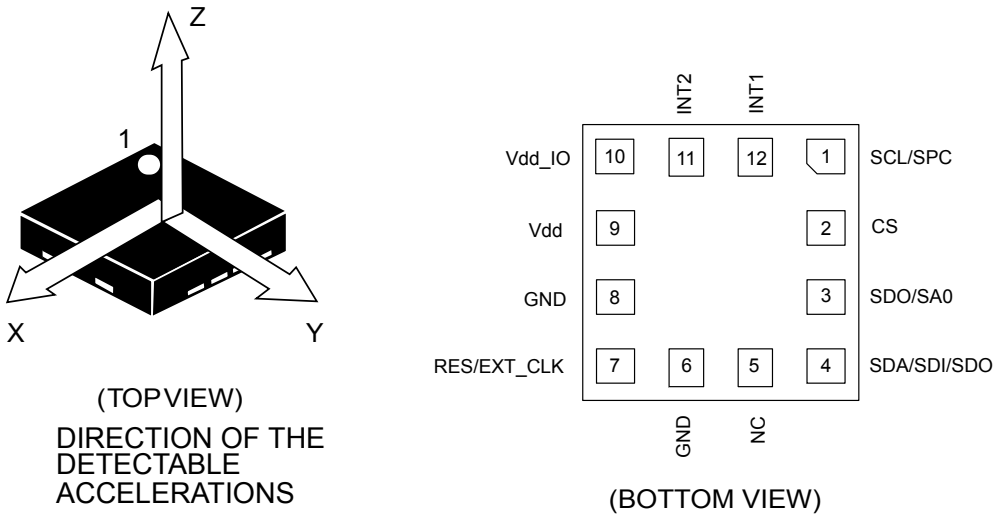


Table 1. Internal pin status

Pin #	Name	Function	Pin status
1	SCL SPC	I²C/MIPI I3C® serial clock (SCL) SPI serial port clock (SPC)	Default: input without internal pull-up
2	CS	SPI/I²C/MIPI I3C® mode selection (1: SPI idle mode / I²C/MIPI I3C® enabled; 0: SPI enabled / I²C/MIPI I3C® disabled)	Default: input with internal pull-up(1)
3	SDO SA0	SPI serial data output (SDO) I²C less significant bit of the device address (SA0)	Default: input without internal pull-up(2)
4	SDA SDI SDO	I²C/MIPI I3C® serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)	Default: (SDA) input without internal pull-up(3)
5	NC	Internally not connected. Can be tied to Vdd, Vdd_IO, or GND.	
6	GND	0 V supply	
7	RES/EXT_CLK	Connect to GND if not used as interrupt pin 1(4) External clock for synchronization of multiple sensors(5)	
8	GND	0 V supply	
9	Vdd	Power supply	
10	Vdd_IO	Power supply for I/O pins	
11	INT2	Interrupt pin 2 Clock input when selected in single data conversion on demand AH input 2 (or Qvar electrode 2) is connected if the analog hub (or Qvar functionality) is enabled.(6)	Default: input with internal pull-down(7)
12	INT1	Interrupt pin 1 AH input 1 (or Qvar electrode 1) is connected if the analog hub (or Qvar functionality) is enabled.(6)	Default: input with internal pull-down(8)

1. *The internal pull-up of the CS pin can be disconnected by setting the CS_PU_DIS bit of register PIN_CTRL (0Ch) to 1.*
2. *The internal pull-up of the SDO/SA0 pin can be connected by setting the SDO_PU_EN bit of register PIN_CTRL (0Ch) to 1.*
3. *The internal pull-up of the SDA/SDI/SDO pin can be connected by setting the SDA_PU_EN bit of register PIN_CTRL (0Ch) to 1.*
4. *When the INT1_ON_RES bit of register CTRL1 (10h) is set to 1, the interrupt signals configured on the INT1 pin are routed to the RES/EXT_CLK pin.*
5. *When the external clock for the synchronization of multiple sensors is intended to be used, the EXT_CLK_EN bit must be set to 1 in register EXT_CLK_CFG (08h) and the bit INT1_ON_RES set to 0 in register CTRL1 (10h) in order to correctly drive the pin.*
6. *The analog hub and Qvar functions are enabled by setting the AH_QVAR_EN bit to 1 in AH_QVAR_CFG (31h).*
7. *The internal pull-down of the INT2 pin can be disconnected by setting the PD_DIS_INT2 bit in PIN_CTRL (0Ch) to 1.*
8. *The internal pull-down of the INT1 pin can be disconnected by setting the PD_DIS_INT1 bit in PIN_CTRL (0Ch) to 1.*

2 Registers

The following registers are accessible when the EMB_FUNC_REG_ACCESS bit in the FUNC_CFG_ACCESS register is set to 0.

Table 2. Main page registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EXT_CLK_CFG	08h	EXT_CLK_EN	0	0	0	0	0	0	0
PIN_CTRL	0Ch	SDO_PU_EN	SDA_PU_EN	PD_DIS_INT2	PD_DIS_INT1	H_LACTIVE	CS_PU_DIS	PP_OD	SIM
WAKE_UP_DUR_EXT	0Eh	0(1)	0(1)	0(1)	WU_DUR_ EXTENDED	0(1)	0(1)	0(1)	0(1)
WHO_AM_I(2)	0Fh	0	1	0	0	0	1	1	1
CTRL1	10h	SMART_POWER_EN	INT1_ON_RES	SW_RESET	IF_ADD_INC	DRDY_PULSED	WU_X_EN	WU_Y_EN	WU_Z_EN
CTRL2	11h	INT1_BOOT	INT1_FIFO_FULL	INT1_FIFO_FTH	INT1_FIFO_OVR	INT1_DRDY	0(1)	0(1)	0(1)
CTRL3	12h	INT2_BOOT	INT2_FIFO_FULL	INT2_FIFO_FTH	INT2_FIFO_OVR	INT2_DRDY	HP_EN	ST_SIGN_X	ST_SIGN_Y
CTRL4	13h	INACT_ODR1	INACT_ODR0	BDU	EMB_FUNC_EN	FIFO_EN	0(1)	SOC	BOOT
CTRL5	14h	ODR3	ODR2	ODR1	ODR0	BW1	BW0	FS1	FS0
FIFO_CTRL	15h	CFG_CHG_EN	FIFO_DEPTH	0(1)	0(1)	STOP_ON_FTH	FIFO_MODE2	FIFO_MODE1	FIFO_MODE0
FIFO_WTM	16h	XL_ONLY_FIFO	FTH6	FTH5	FTH4	FTH3	FTH2	FTH1	FTH0
INTERRUPT_CFG	17h	TIMESTAMP_EN	0(1)	WAKE_THS_W	0(1)	SLEEP_STATUS _ON_INT	DIS_RST_LIR _ALL_INT	LIR	INTERRUPTS _ENABLE
SIXD	18h	D4D_EN	D6D_THS1	D6D_THS0	0(1)	0(1)	0(1)	0(1)	0(1)
WAKE_UP_THS	1Ch	0(1)	SLEEP_ON	WK_THS5	WK_THS4	WK_THS3	WK_THS2	WK_THS1	WK_THS0
WAKE_UP_DUR	1Dh	FF_DUR5	WAKE_DUR1	WAKE_DUR0	ST_SIGN_Z	SLEEP_DUR3	SLEEP_DUR2	SLEEP_DUR1	SLEEP_DUR0
FREE_FALL	1Eh	FF_DUR4	FF_DUR3	FF_DUR2	FF_DUR1	FF_DUR0	FF_THS2	FF_THS1	FF_THS0
MD1_CFG	1Fh	INT1_SLEEP_CHANGE	0(1)	INT1_WU	INT1_FF	INT1_TAP	INT1_6D	INT1_TIMESTAMP	INT1_EMB_FUNC
MD2_CFG	20h	INT2_SLEEP_CHANGE	0(1)	INT2_WU	INT2_FF	INT2_TAP	INT2_6D	INT2_TIMESTAMP	INT2_EMB_FUNC
WAKE_UP_SRC(2)	21h	-	SLEEP_CHANGE_IA	FF_IA	SLEEP_STATE	WU_IA	X_WU	Y_WU	Z_WU
TAP_SRC(2)	22h	TAP_IA	SINGLE_TAP_IA	DOUBLE_TAP_IA	TRIPLE_TAP_IA	-	-	-	-
SIXD_SRC(2)	23h	-	D6D_IA	ZH	ZL	YH	YL	XH	XL
ALL_INT_SRC(2)	24h	-	SLEEP_CHANGE _IA_ALL	D6D_IA_ALL	TRIPLE_IA_ALL	DOUBLE_TAP_ALL	SINGLE_TAP_ALL	WU_IA_ALL	FF_IA_ALL
STATUS(2)	25h	-	-	INT_GLOBAL	-	-	-	-	DRDY
FIFO_STATUS1(2)	26h	FIFO_WTM_IA	FIFO_OVR_IA	-	-	-	-	-	-
FIFO_STATUS2(2)	27h	FSS7	FSS6	FSS5	FSS4	FSS3	FSS2	FSS1	FSS0
OUT_X_L(2)	28h	OUTX7	OUTX6	OUTX5	OUTX4	OUTX3	OUTX2	OUTX1	OUTX0
OUT_X_H(2)	29h	OUTX15	OUTX14	OUTX13	OUTX12	OUTX11	OUTX10	OUTX9	OUTX8
OUT_Y_L(2)	2Ah	OUTY7	OUTY6	OUTY5	OUTY4	OUTY3	OUTY2	OUTY1	OUTY0
OUT_Y_H(2)	2Bh	OUTY15	OUTY14	OUTY13	OUTY12	OUTY11	OUTY10	OUTY9	OUTY8

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OUT_Z_L[2]	2Ch	OUTZ7	OUTZ6	OUTZ5	OUTZ4	OUTZ3	OUTZ2	OUTZ1	OUTZ0
OUT_Z_H[2]	2Dh	OUTZ15	OUTZ14	OUTZ13	OUTZ12	OUTZ11	OUTZ10	OUTZ9	OUTZ8
OUT_T_AH_QVAR_L[2]	2Eh	OUTT7	OUTT6	OUTT5	OUTT4	OUTT3	OUTT2	OUTT1	OUTT0
OUT_T_AH_QVAR_H[2]	2Fh	OUTT15	OUTT14	OUTT13	OUTT12	OUTT11	OUTT10	OUTT9	OUTT8
AH_QVAR_CFG	31h	AH_QVAR_EN	AH_QVAR_ NOTCH_EN	AH_QVAR_ NOTCH_CUTOFF	AH_QVAR_ C_ZIN_1	AH_QVAR_ C_ZIN_0	AH_QVAR_GAIN_1	AH_QVAR_GAIN_0	-
SELF_TEST	32h	-	-	ST1	ST0	-	-	-	T_AH_QVAR_DIS
I3C_IF_CTRL	33h	DIS_DRSTDAA	-	ASF_ON	-	-	-	BUS_ACT_SEL_1	BUS_ACT_SEL_0
EMB_FUNC_STATUS_MAINPAGE[2]	34h	IS_FSM_LC	0	IS_SIGMOT	IS_TILT	IS_STEP_DET	0	0	0
FSM_STATUS_MAINPAGE[2]	35h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
MLC_STATUS_MAINPAGE[2]	36h	0	0	0	0	IS_MLC4	IS_MLC3	IS_MLC2	IS_MLC1
SLEEP	3Dh	0	0	0	0	0	0	0	DEEP_PD
EN_DEVICE_CONFIG	3Eh	-	-	-	-	-	-	-	SOFT_PD
FUNC_CFG_ACCESS[3]	3Fh	EMB_FUNC_ REG_ACCESS	0	0	0	0	0	0	FSM_WR_ CTRL_EN
FIFO_DATA_OUT_TAG[2]	40h	TAG_SENSOR_4	TAG_SENSOR_3	TAG_SENSOR_2	TAG_SENSOR_1	TAG_SENSOR_0	0	0	-
FIFO_DATA_OUT_X_L[2]	41h	D7	D6	D5	D4	D3	D2	D1	D0
FIFO_DATA_OUT_X_H[2]	42h	D15	D14	D13	D12	D11	D10	D9	D8
FIFO_DATA_OUT_Y_L[2]	43h	D7	D6	D5	D4	D3	D2	D1	D0
FIFO_DATA_OUT_Y_H[2]	44h	D15	D14	D13	D12	D11	D10	D9	D8
FIFO_DATA_OUT_Z_L[2]	45h	D7	D6	D5	D4	D3	D2	D1	D0
FIFO_DATA_OUT_Z_H[2]	46h	D15	D14	D13	D12	D11	D10	D9	D8
FIFO_BATCH_DEC	47h	0	0	0	DEC_TS_BATCH_1	DEC_TS_BATCH_0	BDR_XL_2	BDR_XL_1	BDR_XL_0
TAP_CFG0	6Fh	AXIS1	AXIS0	INVERT_T4	INVERT_T3	INVERT_T2	INVERT_T1	INVERT_T0	-
TAP_CFG1	70h	PRE_STILL_THS3	PRE_STILL_THS2	PRE_STILL_THS1	PRE_STILL_THS0	POST_STILL_T3	POST_STILL_T2	POST_STILL_T1	POST_STILL_T0
TAP_CFG2	71h	POST_STILL_T5	POST_STILL_T4	WAIT_T5	WAIT_T4	WAIT_T3	WAIT_T2	WAIT_T1	WAIT_T0
TAP_CFG3	72h	POST_STILL_THS3	POST_STILL_THS2	POST_STILL_THS1	POST_STILL_THS0	LATENCY_T3	LATENCY_T2	LATENCY_T1	LATENCY_T0
TAP_CFG4	73h	WAIT_END_LATENCY	0	PEAK_THS5	PEAK_THS4	PEAK_THS3	PEAK_THS2	PEAK_THS1	PEAK_THS0
TAP_CFG5	74h	TRIPLE_TAP_EN	DOUBLE_TAP_EN	SINGLE_TAP_EN	REBOUND_T4	REBOUND_T3	REBOUND_T2	REBOUND_T1	REBOUND_T0
TAP_CFG6	75h	PRE_STILL_ST3	PRE_STILL_ST2	PRE_STILL_ST1	PRE_STILL_ST0	PRE_STILL_N3	PRE_STILL_N2	PRE_STILL_N1	PRE_STILL_N0
TIMESTAMP0[2]	7Ah	D31	D30	D29	D28	D27	D26	D25	D24
TIMESTAMP1[2]	7Bh	D23	D22	D21	D20	D19	D18	D17	D16
TIMESTAMP2[2]	7Ch	D15	D14	D13	D12	D11	D10	D9	D8
TIMESTAMP3[2]	7Dh	D7	D6	D5	D4	D3	D2	D1	D0

1. *This bit must be set to 0 for the correct operation of the device.*
2. *Read-only register*
3. *When the EMB_FUNC_REG_ACCESS bit is set to 0, the FUNC_CFG_ACCESS register is a read/write register. When the EMB_FUNC_REG_ACCESS bit is set to 1, the FUNC_CFG_ACCESS register is a write-only register.*

2.1 Embedded functions registers

The EMB_FUNC_EN bit in the CTRL4 register enables the embedded functions and must be set to 1 before accessing the embedded functions registers. These registers are accessible when the EMB_FUNC_REG_ACCESS bit in the FUNC_CFG_ACCESS register is set to 1.

Note: When accessing these registers, the content of the FUNC_CFG_ACCESS register cannot be read, but it can be written.

Registers PAGE_SEL, PAGE_ADDRESS, PAGE_VALUE and PAGE_RW are used to configure the desired embedded advance features page and read/write all the embedded advance features registers (see Section 2.2: Embedded advanced features registers).

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(1)	0(1)	0(1)	1(2)
EMB_FUNC_EN_A	04h	MLC_BEFORE_FSM_EN	0(1)	SIGN_MOTION_EN	TILT_EN	PEDO_EN	0(1)	0(1)	0(1)
EMB_FUNC_EN_B	05h	0(1)	0(1)	0(1)	MLC_EN	0(1)	0(1)	0(1)	FSM_EN
EMB_FUNC_EXEC_STATUS(3)	07h	0	0	0	0	0	0	EMB_FUNC_EXEC_OVR	EMB_FUNC_ENDOP
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(1)	INT1_SIG_MOT	INT1_TILT	INT1_STEP_DET	0(1)	0(1)	0(1)
FSM_INT1	0Bh	INT1_FSM8	INT1_FSM7	INT1_FSM6	INT1_FSM5	INT1_FSM4	INT1_FSM3	INT1_FSM2	INT1_FSM1
MLC_INT1	0Dh	0(1)	0(1)	0(1)	0(1)	INT1_MLC4	INT1_MLC3	INT1_MLC2	INT1_MLC1
EMB_FUNC_INT2	0Eh	INT2_FSM_LC	0(1)	INT2_SIG_MOT	INT2_TILT	INT2_STEP_DET	0(1)	0(1)	0(1)
FSM_INT2	0Fh	INT2_FSM8	INT2_FSM7	INT2_FSM6	INT2_FSM5	INT2_FSM4	INT2_FSM3	INT2_FSM2	INT2_FSM1
MLC_INT2	11h	0(1)	0(1)	0(1)	0(1)	INT2_MLC4	INT2_MLC3	INT2_MLC2	INT2_MLC1
EMB_FUNC_STATUS(3)	12h	IS_FSM_LC	0	IS_SIGMOT	IS_TILT	IS_STEP_DET	0	0	0
FSM_STATUS(3)	13h	IS_FSM8	IS_FSM7	IS_FSM6	IS_FSM5	IS_FSM4	IS_FSM3	IS_FSM2	IS_FSM1
MLC_STATUS(3)	15h	0	0	0	0	IS_MLC4	IS_MLC3	IS_MLC2	IS_MLC1
PAGE_RW	17h	EMB_FUNC_LIR	PAGE_WRITE	PAGE_READ	0(1)	0(1)	0(1)	0(1)	0(1)
EMB_FUNC_FIFO_EN	18h	0(1)	0(1)	0(1)	0(1)	FSM_FIFO_EN	MLC_FILTER_FEATURE_FIFO_EN	MLC_FIFO_EN	STEP_COUNTER_FIFO_EN
FSM_ENABLE	1Ah	FSM8_EN	FSM7_EN	FSM6_EN	FSM5_EN	FSM4_EN	FSM3_EN	FSM2_EN	FSM1_EN
FSM_LONG_COUNTER_L	1Ch	FSM_LC_7	FSM_LC_6	FSM_LC_5	FSM_LC_4	FSM_LC_3	FSM_LC_2	FSM_LC_1	FSM_LC_0

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LONG_COUNTER_H	1Dh	-	FSM_LC_14	FSM_LC_13	FSM_LC_12	FSM_LC_11	FSM_LC_10	FSM_LC_9	FSM_LC_8
INT_ACK_MASK	1Fh	IACK_MASK7	IACK_MASK6	IACK_MASK5	IACK_MASK4	IACK_MASK3	IACK_MASK2	IACK_MASK1	IACK_MASK0
FSM_OUTS1(3)	20h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS2(3)	21h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS3(3)	22h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS4(3)	23h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS5(3)	24h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS6(3)	25h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS7(3)	26h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
FSM_OUTS8(3)	27h	P_X	N_X	P_Y	N_Y	P_Z	N_Z	P_V	N_V
STEP_COUNTER_L(3)	28h	STEP_7	STEP_6	STEP_5	STEP_4	STEP_3	STEP_2	STEP_1	STEP_0
STEP_COUNTER_H(3)	29h	STEP_15	STEP_14	STEP_13	STEP_12	STEP_11	STEP_10	STEP_9	STEP_8
EMB_FUNC_SRC(3)	2Ah	PEDO_ RST_STEP	0(1)	STEP_ DETECTED	STEP_COUNT_ DELTA_IA	STEP_ OVERFLOW	STEPCOUNTE R_BIT_SET	0(1)	0(1)
EMB_FUNC_INIT_A	2Ch	MLC_BEFORE_ FSM_INIT	0(1)	SIG_MOT_INIT	TILT_INIT	STEP_DET_ INIT	0(1)	0(1)	0(1)
EMB_FUNC_INIT_B	2Dh	0(1)	0(1)	0(1)	MLC_INIT	0(1)	0(1)	0(1)	FSM_INIT
MLC1_SRC(3)	34h	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(3)	35h	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(3)	36h	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(3)	37h	MLC4_SRC_7	MLC4_SRC_6	MLC4_SRC_5	MLC4_SRC_4	MLC4_SRC_3	MLC4_SRC_2	MLC4_SRC_1	MLC4_SRC_0
FSM_ODR	39h	0(1)	1(2)	FSM_ODR_2	FSM_ODR_1	FSM_ODR_0	0(1)	0(1)	0(1)
MLC_ODR	3Ah	0(1)	MLC_ODR_2	MLC_ODR_1	MLC_ODR_0	0(1)	0(1)	0(1)	1(2)

1. *This bit must be set to 0 for the correct operation of the device.*
2. *This bit must be set to 1 for the correct operation of the device.*
3. *Read-only register*

2.2 Embedded advanced features registers

Embedded advance features registers can be written and read through the procedures described in Section 2.2.1: Write procedure for embedded advanced features registers and Section 2.2.2: Read procedure for embedded advanced features registers.dita.

Note: When accessing these registers by setting either the PAGE_WRITE bit or the PAGE_READ bit of the PAGE_RW register, the content of the FUNC_CFG_ACCESS register cannot be read or written.

Table 4. Embedded advanced features page 0 registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
FSM_LC_TIMEOUT_L	54h	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	55h	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	56h	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	58h	FSM_START7	FSM_START6	FSM_START5	FSM_START4	FSM_START3	FSM_START2	FSM_START1	FSM_START0
FSM_START_ADD_H	59h	FSM_START15	FSM_START14	FSM_START13	FSM_START12	FSM_START11	FSM_START10	FSM_START9	FSM_START8
PEDO_CMD_REG	5Dh	0(1)	0(1)	0(1)	0(1)	CARRY_COUNT_ EN	FP_REJECTION_ EN	0(1)	0(1)
PEDO_DEB_STEPS_CONF	5Eh	DEB_STEP7	DEB_STEP6	DEB_STEP5	DEB_STEP4	DEB_STEP3	DEB_STEP2	DEB_STEP1	DEB_STEP0
PEDO_SC_DELTAT_L	AAh	PD_SC_7	PD_SC_6	PD_SC_5	PD_SC_4	PD_SC_3	PD_SC_2	PD_SC_1	PD_SC_0
PEDO_SC_DELTAT_H	ABh	PD_SC_15	PD_SC_14	PD_SC_13	PD_SC_12	PD_SC_11	PD_SC_10	PD_SC_9	PD_SC_8
T_AH_QVAR_SENSITIVITY_L	B6h	T_AH_QVAR_ S_7	T_AH_QVAR_ S_6	T_AH_QVAR_ S_5	T_AH_QVAR_ S_4	T_AH_QVAR_ S_3	T_AH_QVAR_ S_2	T_AH_QVAR_ S_1	T_AH_QVAR_ S_0
T_AH_QVAR_SENSITIVITY_H	B7h	T_AH_QVAR_ S_15	T_AH_QVAR_ S_14	T_AH_QVAR_ S_13	T_AH_QVAR_ S_12	T_AH_QVAR_ S_11	T_AH_QVAR_ S_10	T_AH_QVAR_ S_9	T_AH_QVAR_ S_8
SMART_POWER_CTRL	D2h	SMART_POWER _CTRL_DUR3	SMART_POWER _CTRL_DUR2	SMART_POWER _CTRL_DUR1	SMART_POWER _CTRL_DUR0	SMART_POWER _CTRL_THS3	SMART_POWER _CTRL_THS2	SMART_POWER _CTRL_THS1	SMART_POWER _CTRL_THS0

1. *This bit must be set to 0 for the correct operation of the device.*

2.2.1 Write procedure for embedded advanced features registers

Write procedure example: write value YYh in the register at address XXh of the embedded advanced features page 0.

1. Write bit EMB_FUNC_REG_ACCESS = 1 in FUNC_CFG_ACCESS (3Fh) // Enable access to the embedded functions registers
2. Write bit PAGE_WRITE = 1 in PAGE_RW (17h) // Select write operation mode
3. Write 0000 in the PAGE_SEL[3:0] field of register PAGE_SEL (02h) // Select page 0
4. Write XXh in PAGE_ADDRESS (08h) // Set address
5. Write YYh in PAGE_VALUE (09h) // Set value to be written
6. Write bit PAGE_WRITE = 0 in PAGE_RW (17h) // Disable write operation mode
7. Write bit EMB_FUNC_REG_ACCESS = 0 in FUNC_CFG_ACCESS (3Fh) // Disable access to the embedded functions registers

2.2.2 Read procedure for embedded advanced features registers.dita

Read procedure example: read the value of the register at address XXh of the embedded advanced features page 0.

1. Write bit EMB_FUNC_REG_ACCESS = 1 in FUNC_CFG_ACCESS (3Fh) // Enable access to the embedded functions registers
2. Write bit PAGE_READ = 1 in PAGE_RW (17h) // Select read operation mode
3. Write 0000 in the PAGE_SEL[3:0] field of register PAGE_SEL (02h) // Select page 0
4. Write XXh in PAGE_ADDRESS (08h) // Set address
5. Read value of PAGE_VALUE (09h) // Get register value
6. Write bit PAGE_READ = 0 in PAGE_RW (17h) // Disable read operation mode
7. Write bit EMB_FUNC_REG_ACCESS = 0 in FUNC_CFG_ACCESS (3Fh) // Disable access to the embedded functions registers

Note: Steps 1 and 2 of both procedures are intended to be performed only at the beginning. Steps 6 and 7 of both procedures are intended to be performed only at the end. If the procedure involves multiple operations, only steps 3, 4 and 5 must be repeated for each operation. Moreover, if a multiple write operation involves consecutive registers, only step 5 must be performed repeatedly.

3 Operating modes

The LIS2DUXS12 provides three possible operating configurations:

- Deep / soft power-down: almost all internal blocks of the device are switched off to minimize power consumption.
- Continuous conversion: the device continuously samples the signal at the data rate selected through the ODR[3:0] bits in the CTRL5 register.
- One-shot: the device waits for a trigger signal in order to generate new data.

All the operating modes of the LIS2DUXS12 can be selected through the ODR[3:0] field in the CTRL5 register.

Table 5. Operating modes

ODR[3:0]	Operating mode
0000	Power-down
0001	1.6 Hz in ultralow-power mode
0010	3 Hz in ultralow-power mode
0011	25 Hz in ultralow-power mode
0100	6 Hz
0101	12.5 Hz
0110	25 Hz
0111	50 Hz
1000	100 Hz
1001	200 Hz
1010	400 Hz
1011	800 Hz
1110	One-shot mode using the INT2 pin
1111	One-shot using the I²C/SPI/MIPI I3C® serial interface (SOC bit in the CTRL4 register)

The device offers a wide Vdd voltage range from 1.62 V to 3.6 V with an independent IO supply from 1.62 V to 3.6 V for the I²C and SPI interfaces and an independent IO supply extended range from 1.08 V to 3.6 V for the MIPI I3C® interface.

3.1 Power-up sequence

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 as floating or connected to ground, until Vdd_IO is set. After Vdd_IO is set, the lines connected to the IO pins have to be configured according to their default status described in Table 1. Internal pin status. In order to avoid an unexpected increase in supply current, the input pins that are not pulled-up/pulled-down must be polarized by the host.

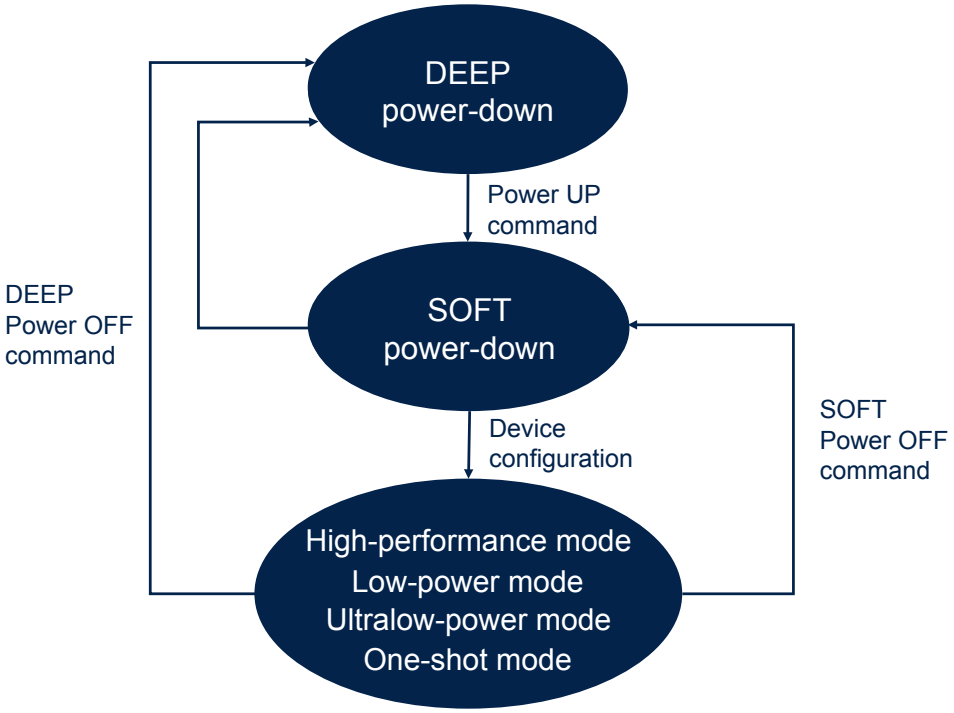
To guarantee proper power-off of the device, it is recommended to maintain the duration of the Vdd line to GND for at least 100 µs.

When Vdd and Vdd_IO are set, the device enters a deep power-down state. This is an extraordinarily low current condition in which the device is powered but with a very aggressive trade-off in supply current.

In order to wake up the device, the first step is to perform a power-up command, bringing the device into a soft power-down state. The device is now ready to be configured and generate data. The power-up command changes depending on which interface is used (see Section 3.1.1: Power-up command).

Note: Unless specified, when "power-down mode" is mentioned in this document, it is meant to be the soft power-down state.

Figure 2. Power-up sequence



When the LIS2DUXS12 is configured in one of the four operating modes, it is possible to switch to the soft power-down condition, writing the ODR[3:0] bits in register CTRL5 (14h) to 0000 (soft power-off command). If the ODR[3:0] bits were set to 1011 (see Table 5. Operating modes), it is necessary to set them to 1010 and wait 3 ms before setting them to 0000. 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 soft power-down mode. It is recommended to switch to soft power-down before switching from one of the continuous conversion modes to one-shot mode (or vice versa). The supply current in soft power-down is around 2.17 µA at Vdd = 1.8 V.

Note: Each write in the CTRL5 register is applied after an ODR period (500 µs if the device is in power-down), therefore, if two consecutive writes in the CTRL5 register have to be performed, they must be separated by an ODR period (or 500 µs if the device is in power-down).

When the device is in soft power-down or it is configured in one of the four operating modes (that is, high-performance, low-power, ultralow-power, or one-shot mode), it is possible to switch to the deep power-down condition, writing the bit DEEP_PD in register SLEEP (3Dh) to 1 (deep power-off command). When a transition to deep power-down occurs, all the registers are reset to their default value, and a new power-up command and device configuration need to be performed. The supply current in deep power-down is around 0.02 µA at Vdd = 1.8 V.

When the device is in deep/soft power-down mode, the digital interfaces (I²C, MIPI I3C®, and SPI) are still active to allow communication with the device.

When the device is in continuous conversion mode, three power modes can be selected: high-performance, low-power, or ultralow-power. In high-performance and low-power mode, the data rates available are from 6 Hz to 800 Hz and the antialiasing filter is active, while in ultralow-power mode, the data rates available are 1.6 Hz, 3 Hz, and 25 Hz and the antialiasing filter is disabled in order to minimize power consumption.

When the device is in one-shot mode, the trigger event for new data generation can be associated to a "hardware" or "software" event. The sample rate can be customized from "one sample when needed" up to 40 Hz (the antialiasing filter is disabled).

3.1.1 Power-up command

The power-up command allows the LIS2DUXS12 to transition from deep power-down to soft power-down and to perform a boot procedure to load the trimming parameters. It differs if either the I²C/I3C interface or the SPI interface is used. Refer to the datasheet for more details on the digital interfaces.

3.1.1.1 I²C/I3C interfaces

If the I²C or I3C interfaces are used, the following sequence should be provided to the device:

- ST/SR+ STATIC ADDRESS+R/W (both R and W sequences are supported)

The device generates a NACK and starts power-up. The operation takes 25 ms (maximum) and once completed, the LIS2DUXS12 is in the SOFT_PD state. It is possible to verify the correct transition in the soft power-down state providing again the power-up command (ST/SR+ STATIC ADDRESS+R/W) and checking the ACK generation from the device.

To guarantee the current execution of the power-up command, the I²C/I3C master should operate at open-drain speed using I²C fast mode plus reference timing.

In the LIS2DUXS12, if the bus is at 1.2 V and the device is in deep power-down, the master should use I²C fast mode timing instead of I²C fast mode plus to perform the power-up command. Once the device is awake, the soft power-down rules apply in the same way as the other voltages.

If the I3C interface is used, a dynamic address should be assigned before starting the configuration of the device. Once the DAA procedure is performed, the dynamic address is stored inside the device and it is maintained if the device returns to the deep power-down state. In this event, no other DAA procedures are needed and the power-up commands can be executed directly using the dynamic address.

3.1.1.2 SPI interface

If the SPI interface is used, the LIS2DUXS12 can move from deep power-down to soft power-down by writing the bit SOFT_PD of register EN_DEVICE_CONFIG (3Eh) to 1. The device starts the power-up and this operation takes 25 ms (maximum). In order to verify that the device has correctly completed the transition to soft power-down, the who_am_I value (expected to be equal to 47h) can be checked by reading register WHO_AM_I (0Fh).

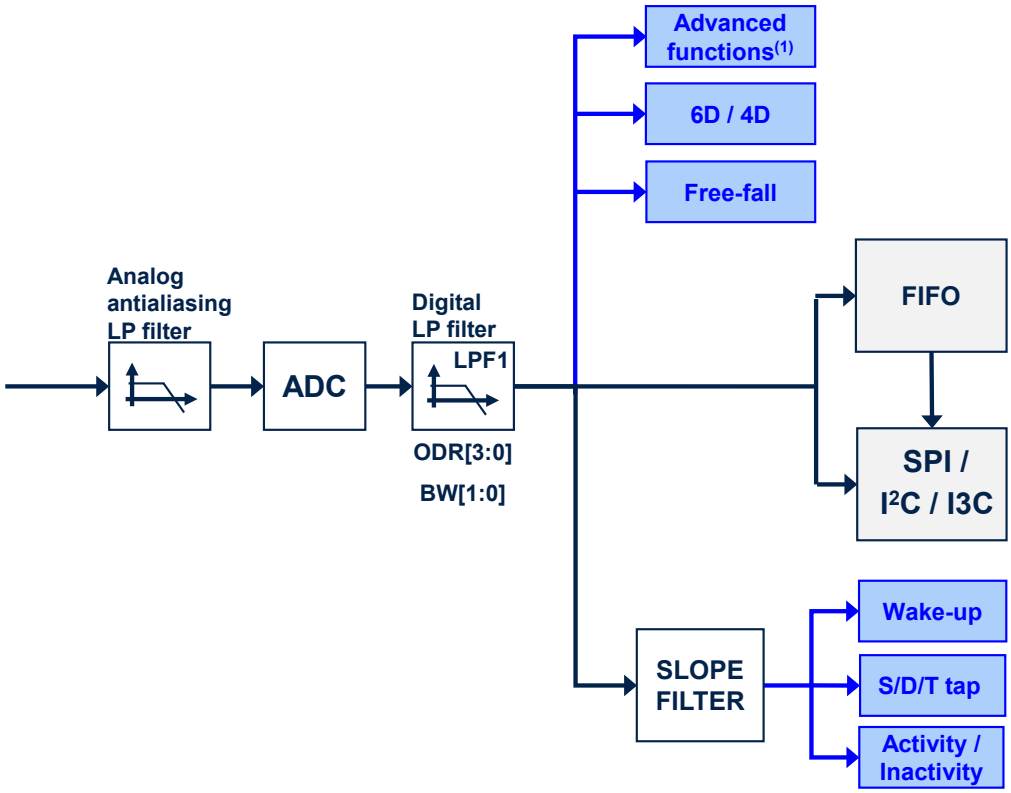
Note: When the power-up command is performed using the SPI interface, the I²C and the I3C interfaces are automatically disabled as soon as the device exits the deep power-down condition. They are automatically re-enabled after a deep power-off command is executed.

3.2 Accelerometer filtering chain

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

Figure 3. Accelerometer filtering chain shows the accelerometer sampling chain. The analog signal coming from the mechanical parts is filtered by an analog antialiasing low-pass filter before being converted by the ADC.

Figure 3. Accelerometer filtering chain

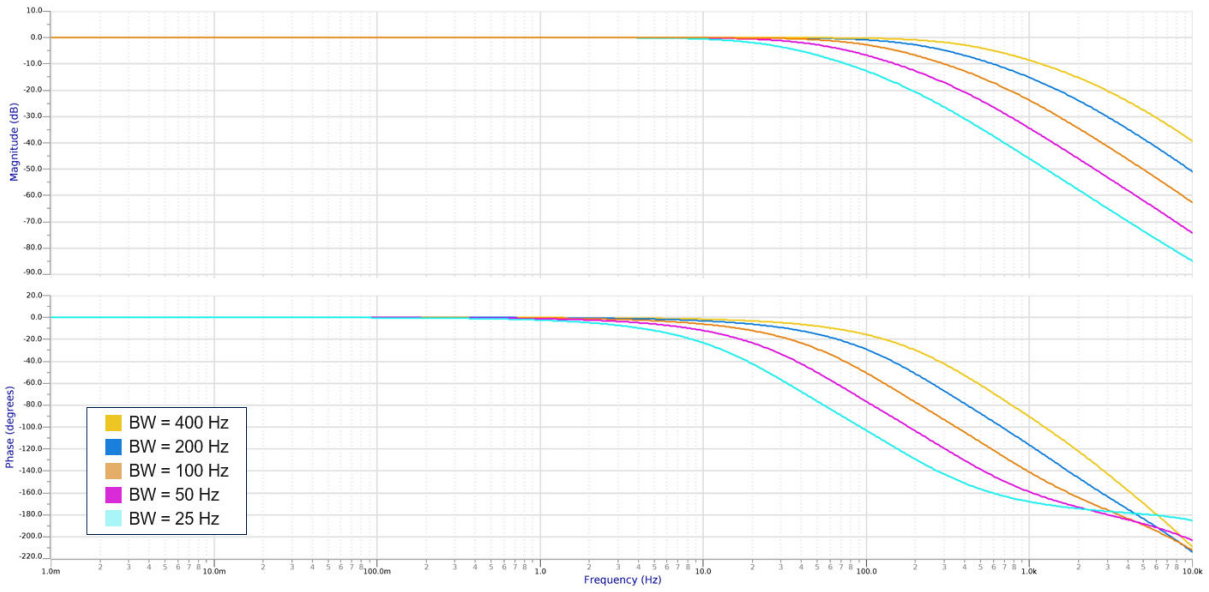


(1) The “Advanced functions” block refers to the pedometer, step detector and step counter, significant motion, and tilt functions; it also includes the finite state machine and the machine learning core.

The analog antialiasing filter is enabled in continuous conversion high-performance mode and low-power mode only. It is disabled in continuous conversion ultralow-power mode and in one-shot mode.

The frequency response (referring to design simulation) of the analog antialiasing filter is shown in Figure 4. The frequency values from 25 Hz to 400 Hz indicated for each curve of the figure refer to the filtering chain bandwidth values (see Table 7 and Table 8 for the complete list of the available cutoff frequencies). For bandwidth values lower than 25 Hz (that is, 12.5 Hz, 6 Hz and 3 Hz), the frequency response is the same as that of 25 Hz.

Figure 4. Antialiasing filter frequency response at different filtering chain bandwidths



The digital LPF1 filter provides different cutoff values based on the accelerometer mode selected, as described in Section 3.3: Continuous conversion and Section 3.4: One-shot.

3.2.1 Accelerometer slope filter

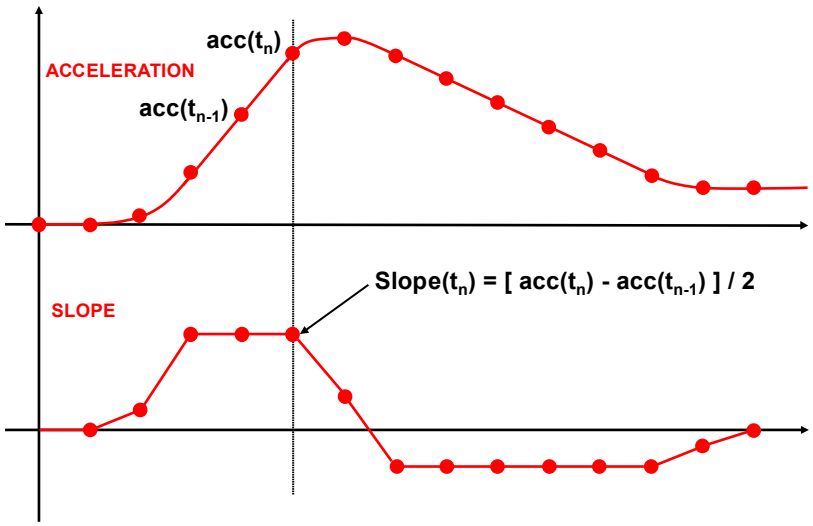
As shown in Figure 5. Accelerometer slope filter, the device embeds a digital slope filter, which can also be used for some embedded features such as single/double/triple-tap recognition, wake-up detection and activity/inactivity.

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 5. Accelerometer slope filter



Note: The first output data generated by the slope filter is not valid and must be discarded.

3.3 Continuous conversion

When one of the continuous conversion data rates is selected through the ODR[3:0] bits in CTRL5, the device continuously samples the signal at the selected data rate.

High-performance mode is enabled/disabled through the HP_EN bit of the CTRL3 (12h) register. If this bit is set to 0 (default value), low-power mode is selected. If it is set to 1, high-performance mode is selected. When selecting ultralow-power mode (by setting the ODR[3:0] bits of the CTRL5 register to the values 0001, 0010 or 0011), the HP_EN bit must be set to 0.

The HP_EN bit should be toggled only when in power-down mode; therefore, direct transitions from low-power to high-performance mode (and vice versa) and from ultralow-power to high-performance (and vice versa) are not possible, as it is necessary to go through power-down mode to toggle the HP_EN bit. Note that after the device has settled in power-down mode, at least 100 µs must elapse before performing another write transaction.

Table 6 provides the maximum settling time in terms of samples to be discarded at the various operating modes. Settling time is defined as the time in which the output reaches 99% of the final value.

Table 6. Accelerometer turn-on/off time

Starting mode	Target mode	Settling time
Power-down	Ultralow-power	None
Power-down	Low-power	See Table 7
Power-down	High-performance	See Table 8
Ultralow-power	Ultralow-power (ODR change)	None
Low-power	Low-power (ODR change)	See Table 7
High-performance	High-performance (ODR change)	See Table 8 + discard 1 additional sample
Ultralow-power	Low-power	See Table 7 + discard 1 additional sample
Low-power	Ultralow-power	None
Low-power / high-performance / ultralow-power	Power-down	2 ms (typical)

3.3.1 Low-power and high-performance modes

When continuous low-power or continuous high-performance mode is set, the filtering chain of LIS2DUXS12, composed of an antialiasing and a low-pass filter, is enabled. The total bandwidth of the sensor can be configured through the BW[1:0] bits in CTRL5.

Table 7. Filtering chain bandwidth in continuous conversion - low-power mode

ODR [Hz]	Cutoff frequency [Hz]	ODR[3:0]	BW[1:0]	Settling time [samples to be discarded]
6	3	0100	11	3
12.5	6	0101	10	3
25	12.5	0110	01	3
50	25	0111	00	3
100	50	1000	00	3
200	100	1001	00	3
400	200	1010	00	2
800	400	1011	00	2
12.5	3	0101	11	6
25	6	0110	10	6
50	12.5	0111	01	6
100	25	1000	01	6
200	50	1001	01	5
400	100	1010	01	4
800	200	1011	01	2
25	3	0110	11	10
50	6	0111	10	10
100	12.5	1000	10	10
200	25	1001	10	9
400	50	1010	10	8
800	100	1011	10	6
50	3	0111	11	20
100	6	1000	11	20
200	12.5	1001	11	20
400	25	1010	11	16
800	50	1011	11	14

Table 8. **Filtering chain bandwidth in continuous conversion - high-performance mode**

ODR [Hz]	Cutoff frequency [Hz]	ODR[3:0]	BW[1:0]	Settling time [samples to be discarded]
6	3	0100	00	2
12.5	6	0101	00	2
25	12.5	0110	00	2
50	25	0111	00	2
100	50	1000	00	2
200	100	1001	00	2
400	200	1010	00	2
800	400	1011	00	2
6	1.5	0100	01	3
12.5	3	0101	01	3
25	6	0110	01	3
50	12.5	0111	01	3
100	25	1000	01	3
200	50	1001	01	3
400	100	1010	01	3
800	200	1011	01	3
6	0.75	0100	10	9
12.5	1.5	0101	10	9
25	3	0110	10	9
50	6	0111	10	9
100	12.5	1000	10	8
200	25	1001	10	8
400	50	1010	10	7
800	100	1011	10	7
6	0.375	0100	11	20
12.5	0.75	0101	11	20
25	1.5	0110	11	20
50	3	0111	11	20
100	6	1000	11	20
200	12.5	1001	11	20
400	25	1010	11	20
800	50	1011	11	18

In continuous low-power or continuous high-performance mode, the entire configurable filtering chain is active, but the overall supply current remains very low.

Table 9. Typical RMS noise and power consumption in continuous conversion - low-power mode [Vdd = 1.8 V, BW_{-3db} = ODR/2]

ODR [Hz]	FS = ±2 *g*		FS = ±4 *g*		FS = ±8 *g*		FS = ±16 *g*	
	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]
6	0.9	8.0	1.2	8.0	2.2	6.2	4.4	5.4
12.5	1.1	8.0	1.3	8.0	2.4	6.2	4.6	5.4
25	1.4	8.0	1.5	8.0	2.8	6.2	5.0	5.4
50	1.9	8.0	1.9	8.0	3.6	6.2	6.5	5.4
100	2.3	8.2	2.3	8.2	3.7	6.4	6.5	5.6
200	2.8	8.6	2.9	8.7	4.2	6.8	6.8	6.0
400	3.5	9.5	3.7	9.6	5.2	7.7	7.6	7.0
800	4.9	10.6	5.2	10.7	6.9	8.8	9.0	8.1

Table 10. Typical RMS noise and power consumption in continuous conversion - high-performance mode [Vdd = 1.8 V, BW_{-3db} = ODR/2]

ODR [Hz]	FS = ±2 *g*		FS = ±4 *g*		FS = ±8 *g*		FS = ±16 *g*	
	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]
6	0.6	14.2	0.5	14.2	0.6	10.8	1.0	9.2
12.5	0.6	14.2	0.6	14.2	0.8	10.8	1.2	9.2
25	0.8	14.2	0.8	14.2	1.1	10.8	1.7	9.2
50	1.1	14.2	1.0	14.2	1.5	10.8	2.3	9.2
100	1.5	14.2	1.4	14.2	2.2	10.8	3.1	9.2
200	2.0	14.2	2.0	14.2	3.0	10.8	4.4	9.2
400	2.7	14.2	2.8	14.2	4.2	10.8	6.1	9.2
800	3.8	14.2	3.9	14.2	5.9	10.8	8.5	9.2

3.3.2 Ultralow-power mode

If the application requires extraordinarily low power consumption and low sampling rate, continuous ultralow-power mode is appropriate. When continuous ultralow-power mode is active, the entire configurable filtering chain is disabled (antialiasing filter also disabled) in order to minimize power consumption, and the cutoff frequency of the system is 400 Hz.

Table 11. Typical RMS noise and power consumption in continuous conversion - ultralow-power mode [Vdd = 1.8 V, BW = ODR/2]

ODR [Hz]	FS = ±2 *g*		FS = ±4 *g*		FS = ±8 *g*		FS = ±16 *g*	
	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]	Noise [mg_{rms}]	Supply current [µA]
1.6	5.3	2.7	4.9	2.8	6.9	2.8	9.1	2.7
3	5.3	2.9	4.8	2.9	7	2.8	9.6	2.8
25	5.2	3.8	5.4	3.9	7.1	3.5	9.5	3.4

3.4 One-shot

If the application requires extraordinary low power consumption but a custom data rate up to 40 Hz is needed, one-shot mode can be used. When one-shot mode is selected, the entire configurable filtering chain is disabled (antialiasing filter also disabled) in order to minimize power consumption, and the total bandwith of the system is 400 Hz.

Figure 6. Single data conversion timing

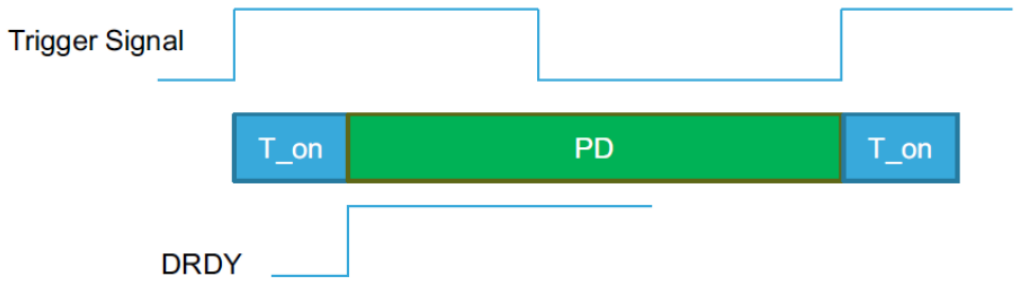


Table 12. Typical turn-on time in one-shot mode using INT2 pin or interface

ODR[3:0]	Operating mode	T_on (typ.)	Max data rate [Hz]
1110	One-shot using the INT2 pin	5.5 ms	40
1111	One-shot using the interface	5.5 ms	40

One-shot mode can use two sampling trigger modes:

- Software: the user starts the acquisition through the I²C / SPI / I3C interface, writing 1 to the SOC bit in the CTRL4 register. The SOC bit in the CTRL4 register is automatically reset when the measurement is finished in order to allow sending a new trigger for the next measurement.
- Hardware: the rising edge signal on the INT2 pin starts a new measurement. The external trigger must be reset to 0 by the user to allow sending a new trigger for the next measurement. The minimum duration for the trigger signal is 100 µs (typ). This mode is not available if the analog hub or Qvar is enabled (refer to Section 9: Analog hub and Qvar sensor).

New data are stored in output registers (28h - 2Fh) when ready.

3.5 External oscillator

The RES/EXT_CLK pin can be used to connect an external oscillator, replacing the internal one, for the synchronization of multiple sensors. The supported clock frequency is 102.4 kHz ± 5% and the supported duty cycle is 50% ± 10%.

When an external oscillator is intended to be used, the EXT_CLK_EN bit in the EXT_CLK_CFG (08h) register must be set to 1 and the INT1_ON_RES bit in the CTRL1 (10h) register must be set to 0 in order to correctly drive the pin.

The result of connecting the same external clock to multiple devices is that all devices will have exactly the same ODR period. More precisely, the samples of these devices will be generated with the same time interval; however, this does not mean that they will be generated at the same time, since the generation time of the first sample depends exclusively on when the start command is issued.

4 Reading output data

4.1 Startup sequence

Once the power-up command is performed, the device automatically downloads the calibration coefficients from the embedded nonvolatile memory to the internal registers. When the boot procedure is completed, that is, after approximately 25 milliseconds, the device automatically enters soft power-down. The default status of the pins with both Vdd and Vdd_IO "on" is indicated in Table 1. Internal pin status.

To turn on the sensors and start gathering data, it is necessary to select one of the operating modes through the ODR[3:0] bits in the CTRL5 register.

Refer to Section 3: Operating modes for a detailed description of data generation.

4.2 Using the status register

The device is provided with a STATUS (25h) register that can be polled to check when a new set of data is available. The DRDY bit is set to 1 when a new set of data is available in the output registers. The read operations should be performed as follows:

1. Read the STATUS register.
2. If DRDY = 0, then go to 1.
3. Read the OUT_X_L register.
4. Read the OUT_X_H register.
5. Read the OUT_Y_L register.
6. Read the OUT_Y_H register.
7. Read the OUT_Z_L register.
8. Read the OUT_Z_H register.
9. Read the OUT_T_AH_QVAR_L register.
10. Read the OUT_T_AH_QVAR_H register.
11. Data processing
12. Go to 1.

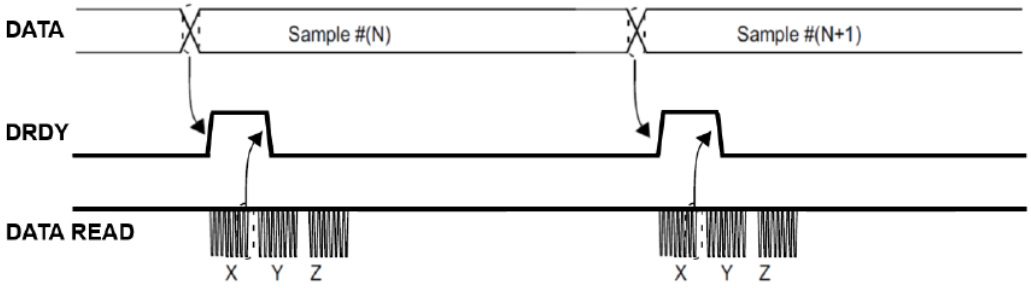
If a multiple-byte read procedure of more than 6 bytes is initiated from the OUT_X_L (28h) register, the procedure automatically returns to the OUT_X_L (28h) register after reading the OUT_Z_H (2Dh) register (wraparound of six registers).

Similarly, if a multiple-byte read procedure of more than 2 bytes is initiated from the OUT_T_AH_QVAR_L (2Eh) register, the procedure automatically returns to the OUT_X_L (28h) register after reading the OUT_T_AH_QVAR_H (2Fh) register (wraparound of eight registers). Therefore, if both accelerometer and Qvar output data (a total of 8 bytes) need to be read in a single multiple-byte read procedure, the procedure must be initiated from the OUT_T_AH_QVAR_L (2Eh) register.

4.3 Using the data-ready signal

The device can be configured to have a hardware signal to determine when a new set of measurement data is available to be read. The data-ready signal is controlled by the DRDY bit of the STATUS register. The signal can be driven to the INT1 pin by setting the INT1_DRDY bit of the CTRL2 register to 1 and to the INT2 pin by setting the INT2_DRDY bit of the CTRL3 register to 1. The data-ready signal rises to 1 when a new set of data has been measured and is available to be read. In DRDY latched mode (bit DRDY_PULSED = 0 in the CTRL1 register), which is the default condition, the signal gets reset when the higher byte of one axis has been read (registers 29h, 2Bh, 2Dh). In DRDY pulsed mode (DRDY_PULSED = 1) the pulse duration is about 90 µs. Pulsed mode is not applied to the DRDY bit, which is always latched.

Figure 7. Data-ready signal



4.4 Using the block data update (BDU) feature

If reading the output data is not synchronized with either the DRDY event bit in the STATUS 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 CTRL4 register.

This feature avoids reading values (most significant and least significant bytes of the output data) related to different samples. In particular, when the BDU is activated, the data registers related to each axis always contain the most recent output data produced by the device, but, in case the read of a given pair (that is, OUT_X_H and OUT_X_L, OUT_Y_H and OUT_Y_L, OUT_Z_H and OUT_Z_L, OUT_T_AH_QVAR_L and OUT_T_AH_QVAR_H) is initiated, the refresh for that pair is blocked until both the MSB and LSB of the data are read.

Note: BDU only guarantees that the LSB and MSB of one data axis (not all data axes) have been sampled at the same moment. For example, if the reading speed is too slow, X and Y can be read at T1 and Z sampled at T2.

The block data update feature is available in ultralow-power mode and low-power mode only, and it can be enabled only if the IF_ADD_INC bit of the CTRL1 register is set to 1.

In high-performance mode, the block data update feature can be implemented through the FIFO buffer (see Section 7: First-in first-out (FIFO) buffer), since the LSB and MSB of each sample are always paired correctly in FIFO. For example, accelerometer data can be read synchronously from FIFO by following this procedure:

1. Set the FIFO_EN bit of the CTRL4 register to 1 to enable batching in FIFO.
2. Set the FTH[6:0] bits of the FIFO_WTM register to 000001 to set the watermark flag value to a single sample.
3. Set the FIFO_MODE[2:0] bits of the FIFO_CTRL register to 011 to enable continuous mode.
4. Set the ODR[3:0] bits of the CTRL5 register to the desired value to start collecting accelerometer data.
5. Poll the FIFO_STATUS1 register until the FIFO_WTM_IA bit is set to 1. The value of this bit can also be routed to the INT1 pin by setting the INT1_FIFO_TH bit to 1 in the CTRL2 register or to the INT2 pin by setting the INT2_FIFO_TH bit to 1 in the CTRL3 register to avoid polling the FIFO_STATUS1 register.
6. Read data from the FIFO output registers (40h-46h): if the tag is related to accelerometer data, parse the output and retrieve the data.
7. Return to step 5 to read the next sample.

4.5 Understanding output data

The measured acceleration data are sent to the OUT_X_H, OUT_X_L, OUT_Y_H, OUT_Y_L, OUT_Z_H, and OUT_Z_L registers. These registers contain, respectively, the most significant byte and the least significant byte of the acceleration signals acting on the X, Y, and Z axes.

The complete output data for the X, Y, Z axes is given by the concatenation OUT_X_H & OUT_X_L, OUT_Y_H & OUT_Y_L, OUT_Z_H & OUT_Z_L.

In ultralow-power and low-power mode, acceleration data are represented (encoded in two's complement) as left-justified 12-bit numbers (the last four digits of the output data of each axis must be ignored).

In high-performance mode, depending on the ODR, acceleration data are represented (encoded in two's complement) as:

- 16-bit numbers at 50 Hz or lower
- Left-justified 15-bit numbers at 100 Hz (the last digit of each axis must be ignored)
- Left-justified 14-bit numbers at 200 Hz (the last two digits of each axis must be ignored)
- Left-justified 13-bit numbers at 400 Hz (the last three digits of each axis must be ignored)
- Left-justified 12-bit numbers at 800 Hz (the last four digits of each axis must be ignored)

Since the numbers are always left-justified, the LSB data obtained through register concatenation can be multiplied by the appropriate sensitivity parameter to obtain the corresponding value in m*g*. The sensitivity parameter only depends on the selected full-scale, as described in the following table.

Table 13. Sensitivity

Full scale	Sensitivity [m*g*/LSB]
±2 *g*	0.061
±4 *g*	0.122
±8 *g*	0.244
±16 *g*	0.488

4.5.1 Example of output data

Below is a simple example of how to use the LSB data and transform it into m*g*. The values are given under the hypothesis of ideal device calibration (that is, no offset, no gain error, and so forth). In this example, the ODR was set to 200 Hz in high-performance mode (useful data on 14 bits) and the full-scale range was set at ±2 *g*.

Get the raw data from the sensor:

```
OUT_X_L: 61h
OUT_X_H: FDh
OUT_Y_L: 73h
OUT_Y_H: 00h
OUT_Z_L: F0h
OUT_Z_H: 42h
```

Do the register concatenation (useful data on 14 bits, last 2 bits must be ignored):

```
(OUT_X_H << 8) + (OUT_X_L & 11111100b) = FD60h [-672 in two's complement]
(OUT_Y_H << 8) + (OUT_Y_L & 11111100b) = 0070h [+112 in two's complement]
(OUT_Z_H << 8) + (OUT_Z_L & 11111100b) = 42F0h [+17136 in two's complement]
```

Apply the sensitivity and obtain the value in m*g* (0.061 m*g*/LSB at full scale ±2 *g*):

```
X: -672 * 0.061 = -41 mg
Y: +112 * 0.061 = +7 mg
Z: +17136 * 0.061 = +1045 mg
```

5 Interrupt generation

In order to generate an interrupt, the LIS2DUXS12 device has to be set in an active operating mode (so, not in power-down) because generation of the interrupts is based on accelerometer data.

The interrupt generator can be configured to detect:

- Free-fall
- Wake-up
- 6D/4D orientation detection
- Single-tap, double-tap, and triple-tap sensing
- Activity/inactivity and motion/stationary recognition

The device can also efficiently run the sensor-related features specified in Android, saving power and enabling faster reaction time. The following functions (called “embedded functions”) are implemented in hardware using only the accelerometer:

- Significant motion
- Relative tilt
- Pedometer functions
- Timestamp

Moreover, the device can be configured to generate interrupt signals activated by user-defined motion patterns. To do this, up to eight embedded finite state machines can be programmed independently for motion detection or gesture recognition such as glance, absolute wrist tilt, shake, double-shake, or pick-up. Furthermore, up to four 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 activity/gesture recognition programs directly provided by STMicroelectronics. Refer to the finite state machine application note and the machine learning core application notes 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.

When the MIPI I3C® interface is used, information about the feature triggering the interrupt event is contained in the in-band interrupt (IBI) frame as described in the datasheet. In this case, the INT1_ON_RES bit of the CTRL1 register must be set to 0.

The H_LACTIVE bit of the PIN_CTRL 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 the interrupt condition is reached.

The PP_OD bit of the PIN_CTRL register allows changing the behavior of the interrupt pins from push-pull to open drain. If the PP_OD bit is set to 0 (default value), the interrupt pins are in push-pull configuration (low-impedance output for both high and low levels). 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 CTRL2 registers for the INT1 pin, and through the MD2_CFG and CTRL3 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 routing a specific interrupt signal to the pin, the related bit has to be set to 1.

Table 14. CTRL2 register

b7	b6	b5	b4	b3	b2	b1	b0
INT1_BOOT	INT1_FIFO _FULL	INT1_ FIFO_TH	INT1_ FIFO_OVR	INT1_DRDY	0	0	0

- INT1_BOOT: boot status 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_DRDY: accelerometer data-ready interrupt on INT1

Table 15. MD1_CFG register

b7	b6	b5	b4	b3	b2	b1	b0
INT1_SLEEP _CHANGE	0	INT1_WU	INT1_FF	INT1_TAP	INT1_6D	INT1_ TIMESTAMP	INT1_ EMB_FUNC

- INT1_SLEEP_CHANGE: activity/inactivity recognition interrupt on INT1
- INT1_WU: wake-up interrupt on INT1
- INT1_FF: free-fall interrupt on INT1
- INT1_TAP: tap recognition interrupt on INT1
- INT1_6D: 6D detection interrupt on INT1
- INT1_TIMESTAMP: interrupt for alert of timestamp overflow within 2.5 ms on INT1
- INT1_EMB_FUNC: embedded functions interrupt on INT1

Table 16. CTRL3 register

b7	b6	b5	b4	b3	b2	b1	b0
INT2_BOOT	INT2_ FIFO_FULL	INT2_ FIFO_TH	INT2_ FIFO_OVR	INT2_DRDY	HP_EN	ST_SIGN_X	ST_SIGN_Y

- INT2_BOOT: boot 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: accelerometer data-ready on INT2

Table 17. **MD2_CFG register**

b7	b6	b5	b4	b3	b2	b1	b0
INT2_SLEEP _CHANGE	0	INT2_WU	INT2_FF	INT2_TAP	INT2_6D	INT2_ TIMESTAMP	INT2_ EMB_FUNC

- INT2_SLEEP_CHANGE: activity/inactivity recognition event interrupt on INT2
- INT2_WU: wake-up interrupt on INT2
- INT2_FF: free-fall interrupt on INT2
- INT2_TAP: tap recognition interrupt on INT2
- INT2_6D: 6D detection interrupt on INT2
- INT2_TIMESTAMP: interrupt for alert of timestamp overflow within 2.5 ms on INT2
- INT2_EMB_FUNC: embedded functions interrupt on INT2

If multiple interrupt signals are routed to the same pin, 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:

- ALL_INT_SRC, WAKE_UP_SRC, TAP_SRC, and SIXD_SRC (for the flags of the basic interrupt functions)
- STATUS (for the data-ready flag through the DRDY bit and for the global flag of the basic interrupt functions through the INT_GLOBAL bit)
- FIFO_STATUS1 (for the FIFO flags)
- EMB_FUNC_STATUS_MAINPAGE / EMB_FUNC_SRC (for the events of the embedded functions)
- FSM_STATUS_MAINPAGE / FSM_STATUS (for the events of the finite state machine)
- MLC_STATUS_MAINPAGE / MLC_STATUS (for the events of the machine learning core)

The ALL_INT_SRC register groups the basic interrupt functions event status (6D/4D, free-fall, wake-up, tap, activity/inactivity) in a single register. It is possible to read this register in order to address a subsequent specific source register read.

The basic interrupts must be enabled by setting the INTERRUPTS_ENABLE bit in the INTERRUPT_CFG register. This bit must be set only when the device is in power-down mode.

The embedded functions must be enabled by setting the EMB_FUNC_EN bit in the CTRL4 register. This action must be performed before configuring the embedded functions.

The LIR bit of the INTERRUPT_CFG 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. Similarly, the EMB_FUNC_LIR bit of the PAGE_RW register enables the latched interrupt request for the embedded functions. In this case, the interrupt signal is reset by reading the EMB_FUNC_STATUS / FSM_STATUS / MLC_STATUS embedded functions register or the EMB_FUNC_STATUS_MAINPAGE / FSM_STATUS_MAINPAGE / MLC_STATUS_MAINPAGE register.

It is possible to disable the reset of the interrupt flags of the basic interrupt functions when reading the ALL_INT_SRC register by setting the DIS_RST_LIR_ALL_INT bit of the INTERRUPT_CFG register. Similarly, it is possible to disable the reset of the interrupt flag of one of the embedded functions when reading the EMB_FUNC_STATUS / FSM_STATUS / MLC_STATUS embedded functions register or the EMB_FUNC_STATUS_MAINPAGE / FSM_STATUS_MAINPAGE / MLC_STATUS_MAINPAGE register by setting the corresponding bit from the IACK_MASK[7:0] bit range of the INT_ACK_MASK embedded functions register.

Note: The embedded functions interrupt is generated about 150 µs before the update of the output data registers and the generation of the DRDY signal. For this reason, if the user reads the output data registers synchronously with the occurrence of the embedded functions interrupt, the data that is read may have not been updated yet. To avoid this misalignment, the user should wait 150 µs or wait for the DRDY signal before reading the output data registers.

The LIR bit in the INTERRUPT_CFG register does not impact the behavior of the DRDY signal. The data-ready signal can be configured to pulsed mode or latched mode based on the value of the DRDY_PULSED bit in the CTRL1 register.

5.2 Wake-up interrupt

The wake-up interrupt signal is generated if a certain number of consecutive accelerometer data exceed the configured threshold (Figure 8. Wake-up event recognition). The wake-up feature is fed by an internal slope filter.

Wake-up recognition can be enabled separately for each axis by using the WU_X_EN, WU_Y_EN, and WU_Z_EN bits in the CTRL1 register: for the generation of a wake-up interrupt, the modulus of the accelerometer data of at least one of the enabled axes must be greater than the threshold parameter for a number of samples at least equal to the duration parameter.

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 INTERRUPT_CFG register:

- If WAKE_THS_W = 0 (default), 1 LSB = FS_XL / 2^6.
- If WAKE_THS_W = 1, 1 LSB = FS_XL / 2^8.

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 a duration of 1/ODR (also known as ODR_time), where ODR is the accelerometer output data rate.

By setting the WU_DUR_EXTENDED bit in the WAKE_UP_DUR_EXT register, the following durations are selectable, instead, by using the WAKE_DUR[1:0] bits of the WAKE_UP_DUR register:

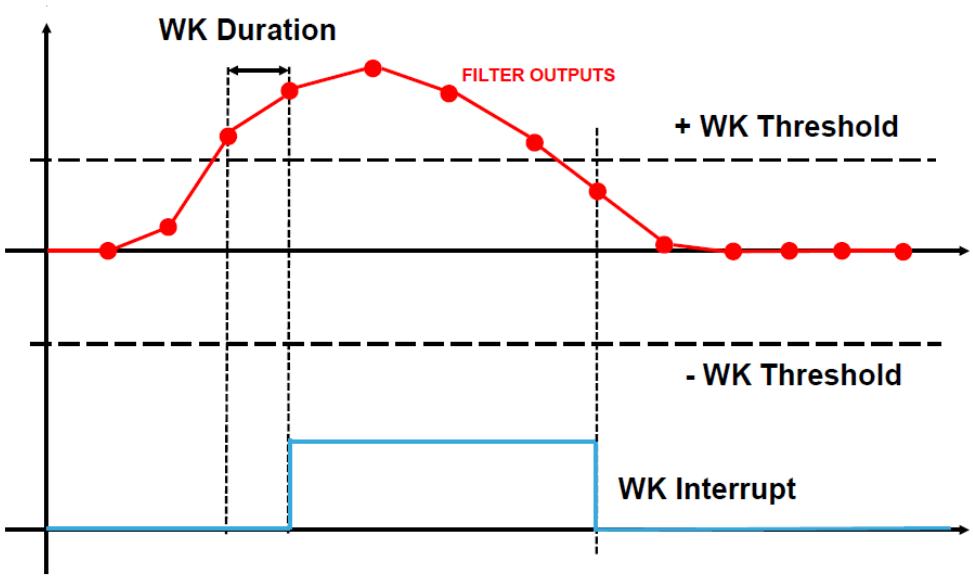
- 00: 3 ODR_time
- 01: 7 ODR_time
- 10: 11 ODR_time
- 11: 15 ODR_time

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 driven to the INT1 (or INT2) pin by setting the INT1_WU (or INT2_WU) bit of the MD1_CFG (or MD2_CFG) register to 1. It can also be checked by reading the WU_IA bit of the WAKE_UP_SRC register or WU_IA_ALL of the ALL_INT_SRC register. The X_WU, Y_WU, and Z_WU bits of the WAKE_UP_SRC register indicate which axis/axes has/have triggered the wake-up event.

If interrupt level mode is enabled, the interrupt signal is automatically reset when the filtered data falls below the threshold. If latched mode is enabled, once a wake-up event has occurred and the interrupt pin is asserted, it must be reset by reading the ALL_INT_SRC register or the WAKE_UP_SRC register. The X_WU, Y_WU, Z_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 X_WU, Y_WU and Z_WU bits have to be evaluated (in addition to the WU_IA bit), it is recommended to read the WAKE_UP_SRC register before reading the ALL_INT_SRC register (see Section 5.1: Interrupt pin configuration for more details on how to configure the interrupt mode).

Figure 8. Wake-up event recognition



The example code that implements the software routine for wake-up event recognition is given below.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write 17h in CTRL1 // Enable wake-up event detection on the X, Y, and Z axes
3. Write 40h in WAKE_UP_DUR // Set wake-up duration
4. Write 02h in WAKE_UP_THS // Set wake-up threshold
5. Write 20h in MD1_CFG // Wake-up interrupt driven to INT1 pin
6. Write 01h in INTERRUPT_CFG // Enable basic interrupts
7. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

In this example, the WU_THS field of the WAKE_UP_THS register is set to 000010, therefore the wake-up threshold is 62.5 m*g* (= 2 * FS / 64).

Since the wake-up functionality is implemented using the slope filter, it is necessary to consider the settling time of this filter if the functionality is enabled. When using the slope filter, the wake-up functionality is based on the comparison of the threshold value with half of the difference between the acceleration data [x,y,z] of the current sample and the acceleration data of the previous one (refer to Section 3.2.1: Accelerometer slope filter). At the very first sample, the slope filter output [x_f,y_f,z_f] is calculated as half of the difference between the first sample (for example, [x,y,z]=[0,0,1*g*]) and an initialization sample with values [x,y,z]=[0,0,0], since a previous sample does not exist yet. For this reason, the first output value of the slope filter (for example, [x_f,y_f,z_f]=[0,0,500m*g*]) could be higher than the threshold value on one of the axes (for example, on the Z-axis), in which case a spurious interrupt event is generated. In order to avoid this spurious interrupt generation, multiple solutions are possible. Three alternative solutions are presented below:

a. ignore the first generated wake-up interrupt signal

b. add a wait time higher than 1 ODR_time before driving the interrupt signal to the INT1/INT2 pin (the status of the interrupt pin must be reset and the eventual wake-up spurious events must be cleared before driving the interrupt signal)

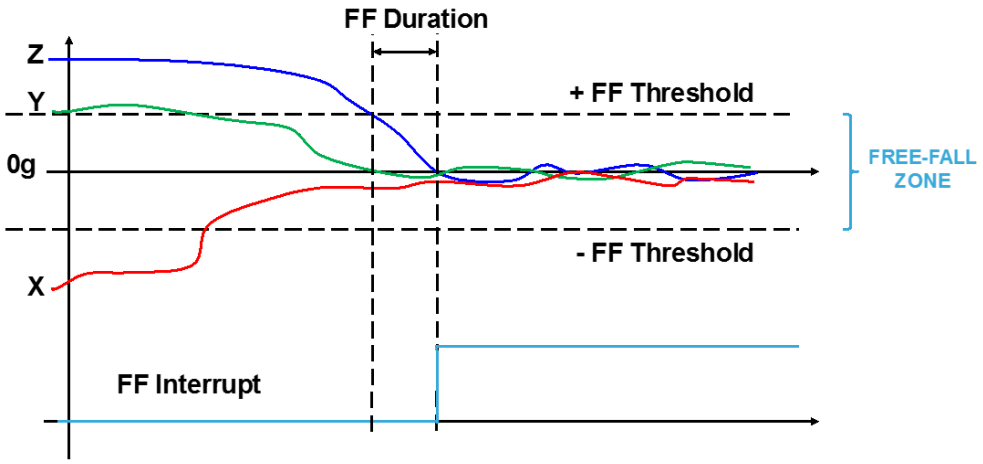
c. set the duration parameter of the wake-up feature to at least 1 ODR_time; then, if needed, set it to 0 after 1 ODR_time

The settling time in terms of samples to be discarded when changing the operating mode of the device must be considered, too, in order to avoid spurious interrupt generation. Refer to Section 3.3: Continuous conversion for the settling time at the various operating modes.

5.3 Free-fall interrupt

Free-fall detection refers to a specific register configuration that allows recognizing when the device is in free-fall: the acceleration measured along all the axes goes to zero. In a real application, a “free-fall zone" is defined around the zero-*g* level where all the accelerations are small enough to generate the interrupt. Configurable threshold and duration parameters are associated to free-fall event detection: the threshold parameter defines the free-fall zone amplitude; the duration parameter defines the minimum duration of the free-fall interrupt event to be recognized (Figure 9. Free-fall interrupt).

Figure 9. Free-fall interrupt



The free-fall event signal can be routed to the INT1(or INT2) pin by setting the INT1_FF (or INT2_FF) bit of the MD1_CFG (or MD2_CFG) register to 1; it can also be checked by reading the FF_IA bit of the WAKE_UP_SRC register or FF_IA_ALL of the ALL_INT_SRC register.

If interrupt level mode is enabled, the interrupt signal is automatically reset when the free-fall condition is no longer verified. If latched mode is enabled, once a free-fall event has occurred and the interrupt pin is asserted, it must be reset by reading the FF_IA bit of the WAKE_UP_SRC register or the FF_IA_ALL bit of the ALL_INT_SRC register (see Section 5.1: Interrupt pin configuration for more details on how to configure the interrupt mode).

The free-fall detection parameters can be modified by configuring the FREE_FALL (contains bits FF_THS[2:0] and FF_DUR[4:0]) and WAKE_UP_DUR (contains MSB of duration parameter - FF_DUR5) registers. The threshold value can be set through the FF_THS[2:0] bits and is described in Table 18. Free-fall threshold values. The values given in this table are valid for any accelerometer full-scale configuration.

Table 18. Free-fall threshold values

FREE_FALL register - FF_THS[2:0]	Threshold value
000	~156 mg
001	~219 mg
010	~250 mg
011	~312 mg
100	~344 mg
101	~406 mg
110	~469 mg
111	~500 mg

A basic software routine for free-fall event recognition is given below.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write 00h in WAKE_UP_DUR // Set event duration (FF_DUR5 = 0)
3. Write 33h in FREE_FALL // Set event duration and threshold
4. Write 10h in MD1_CFG // FF interrupt driven to INT1 pin
5. Write 01h in INTERRUPT_CFG // Enable basic interrupts
6. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

The sample code exploits a threshold set to ~312 m*g* for free-fall recognition and the event is notified by hardware through the INT1 pin. The FF_DUR[5:0] field of the FREE_FALL / WAKE_UP_DUR registers is configured to ignore events that are shorter than 6/ODR = 6/200 Hz = 30 ms in order to avoid false detections.

5.4 6D/4D orientation detection

The LIS2DUXS12 device provides the capability to detect the orientation of the device in space, enabling easy implementation of energy-saving procedures and automatic image rotation for mobile devices.

5.4.1 6D orientation detection

Six orientations of the device in space can be detected; the interrupt signal is asserted when the device switches from one orientation to another. The interrupt is not reasserted as long as the position is maintained. 6D interrupt is generated when only one axis exceeds a selected threshold and the acceleration values measured from the other two axes are lower than the threshold: the ZH, ZL, YH, YL, XH, XL bits of the SIXD_SRC register indicate which axis has triggered the 6D event.

In more detail:

Table 19. SIXD_SRC register

b7	b6	b5	b4	b3	b2	b1	b0
-	D6D_IA	ZH	ZL	YH	YL	XH	XL

The D6D_IA bit is set high when the device switches from one orientation to another. Moreover:

- ZH/YH/XH is set high when the face perpendicular to the Z/Y/X axis is almost flat and the acceleration measured on the Z/Y/X axis is positive and in the absolute value bigger than the threshold.
- ZL/YL/XL is set high when the face perpendicular to the Z/Y/X axis is almost flat and the acceleration measured on the Z/Y/X axis is negative and in the absolute value bigger than the threshold.

The D6D_THS[1:0] bits of the SIXD register are used to select the threshold value used to detect the change in device orientation. The threshold values given in Table 20. Threshold for 4D/6D function are valid for each accelerometer full-scale value.

Table 20. Threshold for 4D/6D function

D6D_THS[1:0]	Threshold value [degrees]
00	80
01	70
10	60
11	50

This interrupt signal can be driven to the INT1 (or INT2) pin by setting the INT1_6D (or INT2_6D) bit of the MD1_CFG (or MD2_CFG) register to 1; it can also be checked by reading the D6D_IA bit of the SIXD_SRC register or the D6D_IA_ALL bit of the ALL_INT_SRC register.

If interrupt level mode is enabled, the interrupt signal is active only for a duration of 1/ODR then it is automatically deasserted (ODR is the accelerometer output data rate). If latched mode is enabled, once an orientation change has occurred and the interrupt pin is asserted, a read of the D6D_IA bit of the SIXD_SRC register or of the D6D_IA_ALL bit of the ALL_INT_SRC register clears the request and the device is ready to recognize a different orientation. The XL, XH, YL, YH, ZL, ZH bits are not affected by the interrupt mode configuration: they correspond to the current state of the device when the SIXD_SRC register is read (see Section 5.1: Interrupt pin configuration for more details on how to configure the interrupt mode).

Referring to the six possible cases illustrated in Figure 10. 6D recognized orientations, the content of the SIXD_SRC register for each position is shown in Table 21. SIXD_SRC register for 6D positions.

Figure 10. 6D recognized orientations

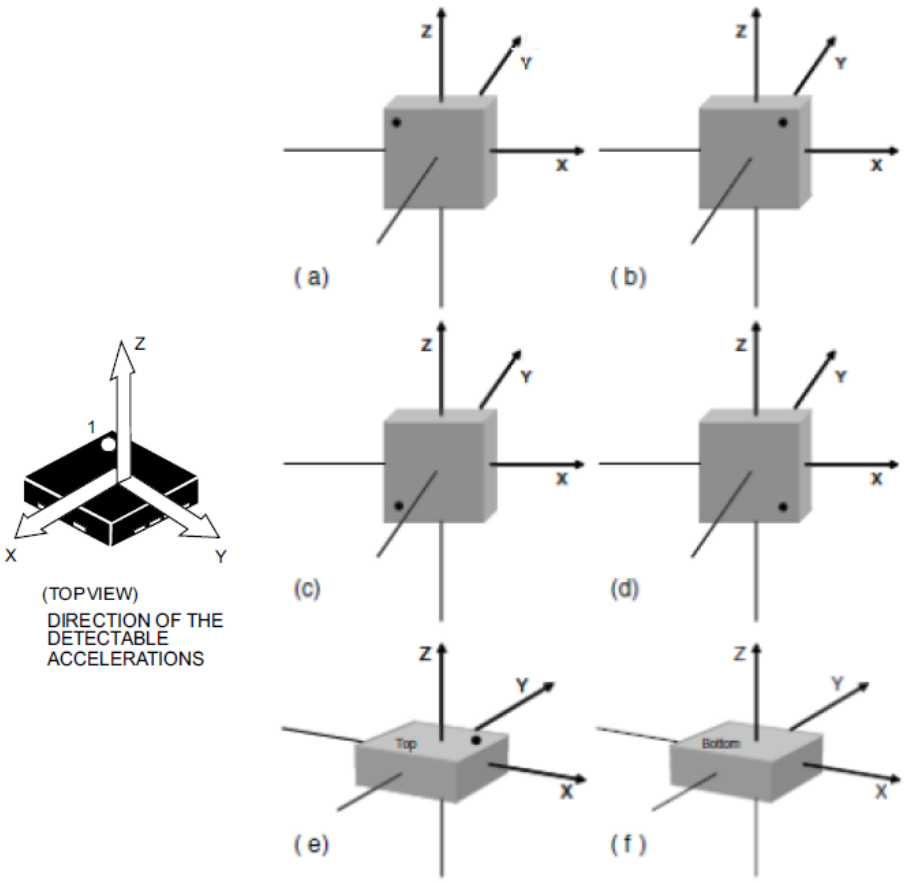


Table 21. SIXD_SRC register for 6D positions

Case	D6D_IA	ZH	ZL	YH	YL	XH	XL
(a)	1	0	0	0	0	0	1
(b)	1	0	0	0	1	0	0
(c)	1	0	0	1	0	0	0
(d)	1	0	0	0	0	1	0
(e)	1	1	0	0	0	0	0
(f)	1	0	1	0	0	0	0

The following example implements a software routine for 6D orientation detection:

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write 40h in SIXD // Set the 6D threshold (D6D_THS[1:0] = 10 = 60 degrees)
3. Write 04h in MD1_CFG // 6D interrupt driven to INT1 pin
4. Write 01h in INTERRUPT_CFG // Enable basic interrupts
5. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

5.4.2 4D orientation detection

The 4D direction function is a subset of the 6D function especially defined to be implemented in mobile devices for portrait and landscape computation. It can be enabled by setting the D4D_EN bit of the SIXD register.

In this configuration, Z-axis position detection is disabled, therefore reducing position recognition to cases (a), (b), (c), and (d) of Table 21. SIXD_SRC register for 6D positions.

5.5 Single-tap, double-tap, and triple-tap recognition

The single-tap, double-tap, and triple-tap recognition functions featured in the LIS2DUXS12 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 along the configured direction.

If the sensor is exposed to a single input stimulus, it generates an interrupt request on the interrupt pin INT1 / INT2. A more advanced feature allows the generation of an interrupt request when a double or triple input stimulus with programmable time between the events is recognized. In the LIS2DUXS12 device, the tap recognition functions use the slope filtered acceleration data as input to detect tap events.

This function can be fully programmed by the user in terms of the expected amplitude and timing of the internal slope filtered data by means of a dedicated set of registers (from TAP_CFG0 to TAP_CFG6).

The recommended accelerometer ODR for tap recognition is 400 Hz or higher.

5.5.1 Single-tap configuration

The signal of a tap event has a well-defined structure, which consists of: a pre-shock stationary phase, a shock phase consisting of two peaks of opposite sign, and a post-shock stationary phase. The flow of a tap event detection is composed of the following main steps:

1. Peak detection
2. Pre-shock stationary condition check
3. Inverted peak detection
4. Post-shock stationary condition check, after a wait time

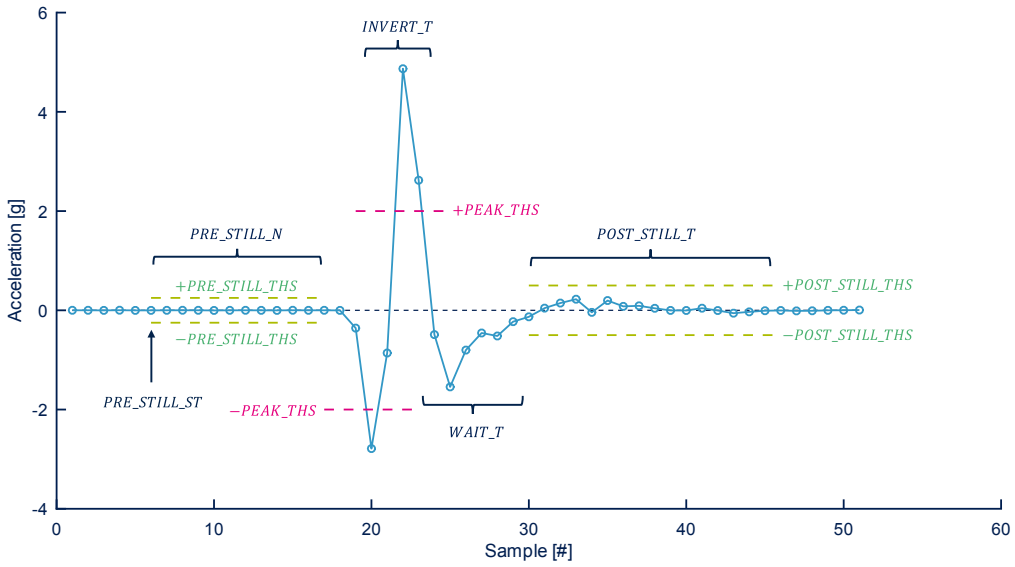
The device is configured for single-tap events detection by setting the SINGLE_TAP_EN bit of the TAP_CFG5 register to 1. The AXIS[1:0] bits of the TAP_CFG0 register allow the selection of the axis for tap event detection with the following logic:

- 00: no axis (default)
- 01: X-axis
- 10: Y-axis
- 11: Z-axis

If the device is configured for single-tap event detection, an interrupt is generated when the following conditions are met (see Figure 11 for reference):

1. The internal slope filtered data must have a peak (either positive or negative) above a threshold configurable through the PEAK_THS[5:0] bits of the TAP_CFG4 register (1 LSB = 62.5 m*g*, maximum 3937.5 m*g*).
2. A configurable number of samples before the peak are in absolute value below the threshold for the pre-shock stationary condition, except for the most recent samples, as they are already part of the shock. This threshold is configurable through the PRE_STILL_THS[3:0] bits of the TAP_CFG1 register (1 LSB = 62.5 m*g*, maximum 937.5 m*g*).
3. The inverted peak (a peak above the threshold defined by PEAK_THS, in the opposite sign of the first peak) is detected within a programmed number of samples from the first peak.
4. A configurable number of samples after the inverted peak are in absolute value below the threshold for post-shock stationary condition, except for the samples immediately after, as they are still part of the shock. This threshold is configurable through the POST_STILL_THS[3:0] bits of the TAP_CFG3 register (1 LSB = 62.5 m*g*, maximum 937.5 m*g*).

Figure 11. Tap event recognition



Timing parameters can be configured with the following:

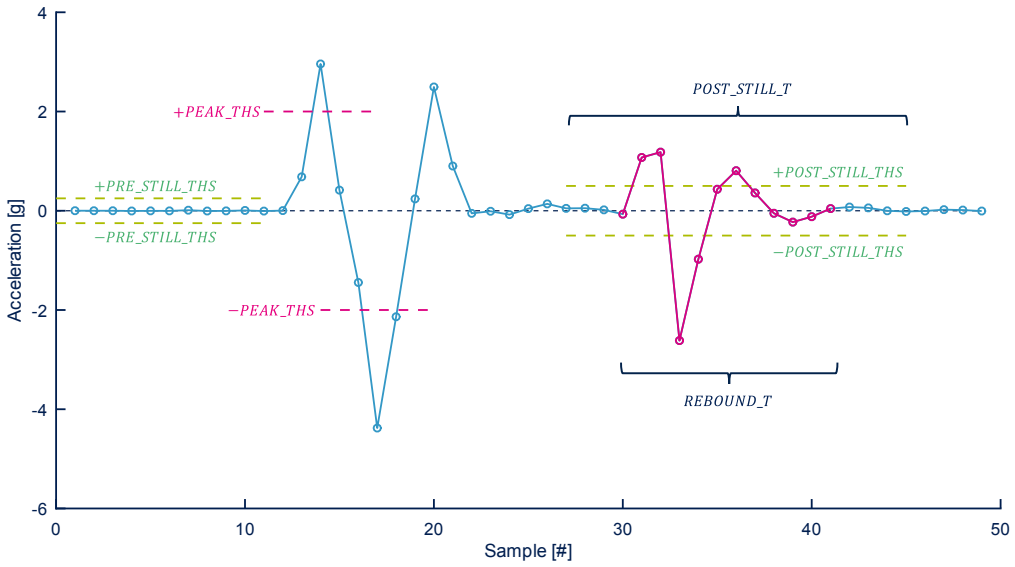
- PRE_STILL_ST[3:0] and PRE_STILL_N[3:0] bits of the TAP_CFG6 register configure the samples to be checked for the stationary condition before the shock. PRE_STILL_ST[3:0] bits select the start sample (from the oldest sample in a buffer of 14 samples before the first peak, ranging from 0 to 13 with 1 LSB = 1 sample), while PRE_STILL_N[3:0] bits select the number of samples to be checked (1 LSB = 1 sample, up to 14 samples). It is mandatory to configure PRE_STILL_ST[3:0] bits and PRE_STILL_N[3:0] bits by respecting the following condition: PRE_STILL_ST + PRESTILL_N ≤ 14. If the value of PRE_STILL_N is set to 0, the pre-shock stationary condition check is disabled.
- INVERT_T[4:0] bits of the TAP_CFG0 register configure the maximum number of samples that can elapse between the first and the second (inverted) peak (1 LSB = 1 sample, maximum 31 samples). If set to 0, the inverted peak detection is disabled.
- WAIT_T[5:0] bits of the TAP_CFG2 register configure the number of samples to wait for the shock to finish before starting to check for the post-shock stationary condition (1 LSB = 2 samples, maximum 126 samples).
- POST_STILL_T[5:0] bits found in the TAP_CFG1 and TAP_CFG2 registers configure the number of samples to be checked for the stationary condition after the shock and wait phases (1 LSB = 4 samples, maximum 252 samples).

In some applications (for example, in hearable devices) rebound is often observed during the post-shock stationary phase (see Figure 12). The device can be configured to manage it. If a sample in the interval defined by POST_STILL_T (that is, the window in which to check the stationary condition after the inverted peak and the wait period) is detected above the POST_STILL_THS threshold, the system does not invalidate the tap detection, and waits for a number of samples for the rebound to finish without checking the post-shock stationary condition. The number of samples to wait can be configured with the REBOUND_T[4:0] bits of the TAP_CFG5 register (1 LSB = 2 samples, maximum 62 samples). The system then checks if the post-shock stationary condition is satisfied for the remaining samples, which is the number of samples defined by the following formula:

$$POST\_STILL\_T - N - REBOUND\_T$$

where N is the number of samples detected correctly in the post-shock stationary phase before the sample overthreshold. Only one rebound event can happen in the window of the post-shock stationary condition for the tap event to be recognized as such. By setting REBOUND_T to 0, the rebound recognition logic is disabled.

Figure 12. Tap event recognition with rebound



5.5.2 Double-tap and triple-tap configuration

The device can be configured for double-tap and triple-tap detection by setting, respectively, the DOUBLE_TAP_EN and TRIPLE_TAP_EN bits of the TAP_CFG5 register to 1.

If the device is configured for double-tap or triple-tap detection, when one or two taps are detected after the first tap, they are recognized as double-tap and triple-tap events instead of consecutive single-tap events. The recognition of double-tap and triple-tap events occurs only if the second and third taps satisfy the rules defined by a latency window. In particular, the LATENCY_T[3:0] bits of the TAP_CFG3 register configure the maximum number of samples that can elapse between consecutive taps in order for them to be recognized as a double-tap or triple-tap event (1 LSB = 32 samples, minimum 16 samples, maximum 480 samples).

5.5.3 Tap recognition interrupts

The tap interrupt signals can be checked by reading the TAP_SRC register:

- The TAP_IA bit is set to 1 when an enabled tap event has been detected.
- The SINGLE_TAP_IA bit is set to 1 when a single-tap tap event (if enabled) has been detected.
- The DOUBLE_TAP_IA bit is set to 1 when a double-tap tap event (if enabled) has been detected.
- The TRIPLE_TAP_IA bit is set to 1 when a triple-tap tap event (if enabled) has been detected.

The WAIT_END_LATENCY bit of the TAP_CFG4 register enables the additional feature of waiting for the end of the latency window to exclusively determine if the event is a single, double, or triple-tap (refer to the following figures). In case WAIT_END_LATENCY is set to 0, the tap event flag is raised immediately for every detected tap. In case WAIT_END_LATENCY is set to 1, the tap event flag is raised immediately if the highest level of enabled tap recognition is reached, otherwise it is raised at the end of the latency window if no additional taps are detected within the window.

Figure 13. Tap recognition flags (WAIT_END_LATENCY = 0)

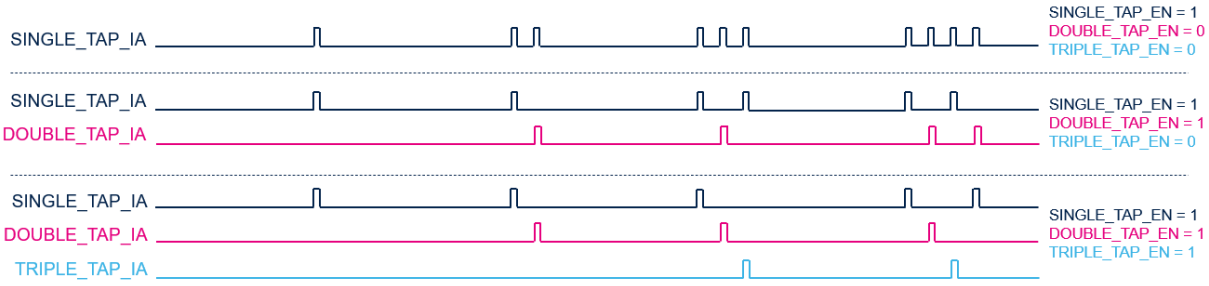
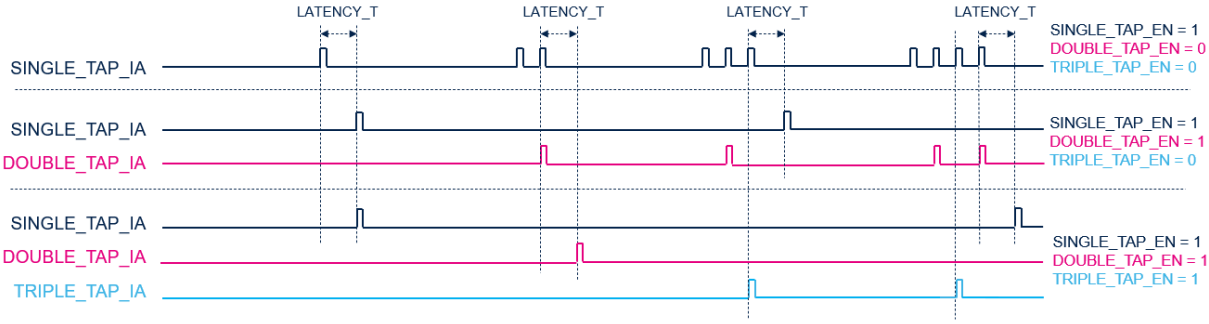


Figure 14. Tap recognition flags (WAIT_END_LATENCY = 1)



The TAP_IA status bit can be routed to the INT1 or to the INT2 pin as an interrupt signal. The INT1_TAP bit of the MD1_CFG register routes the TAP_IA signal to the INT1 pin, while the INT2_TAP bit of the MD2_CFG register routes the TAP_IA signal to the INT2 pin.

If interrupt level mode is enabled, the interrupt is kept high for a time equivalent to 1/ODR. If latched mode is enabled, the interrupt is kept high until either the ALL_INT_SRC or the TAP_SRC register is read.

5.5.4 Tap recognition example

The following procedure describes the steps for configuring single-tap, double-tap, and triple-tap detection. It enables the feature of waiting for the end of the latency window to discriminate the type of the tap event. Rebound recognition is disabled.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write C8h in TAP_CFG0 // Tap event detection on Z-axis, window length for the search of the inverted peak of 4 samples
3. Write 28h in TAP_CFG1 // Threshold for pre-shock stationary condition of 125 mg, window length for post-shock stationary condition of 32 samples
4. Write 03h in TAP_CFG2 // Window length for wait condition of 6 samples
5. Write 84h in TAP_CFG3 // Threshold for post-shock stationary condition of 500 m*g*, latency window length for latency of 128 samples
6. Write 88h in TAP_CFG4 // Enables latency feature, peak detection threshold of 500 mg
7. Write E0h in TAP_CFG5 // Enables single-tap, double-tap, and triple-tap events, no rebound recognition
8. Write 0Ah in TAP_CFG6 // Starting sample of window for pre-shock stationary condition is the 1st sample of the 14-sample buffer window length for pre-shock stationary condition of 10 samples
9. Write 08h in MD1_CFG // Routes TAP_IA to INT1 pin
10. Write 01h in INTERRUPT_CFG // Enables basic interrupts
11. Write A2h in CTRL5 // ODR = 400 Hz, FS ±8 *g*

5.6 Activity/inactivity recognition

The activity/inactivity recognition function allows reducing system power consumption and developing new smart applications.

The activity/inactivity function is enabled by setting the SLEEP_ON bit of the WAKE_UP_THS register to 1. This bit must be set only when the device is in power-down mode. If the sleep state condition is detected, the LIS2DUXS12 automatically goes to the ultralow-power ODR selected by the INACT_ODR[1:0] bits in the CTRL4 register. The LIS2DUXS12 wakes up from the sleep state as soon as a wake-up event has been detected, switching to the operating mode and ODR configured in the CTRL5 register. This function can be enabled only if the HP_EN bit of the CTRL3 register is set to 0 (that is, the device is in low-power mode).

With this feature the system may be efficiently switched from low-power consumption to full performance and vice versa depending on user-selectable acceleration events, thus ensuring power saving and flexibility.

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

The unsigned threshold value is defined using the WK_THS[5:0] bits in 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 INTERRUPT_CFG register:

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

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

When a certain number of consecutive X,Y, Z slope filtered data is smaller than the configured threshold, the ODR[3:0] bits of the CTRL5 register are bypassed (inactivity) and the accelerometer is internally set to the ultralow-power ODR selected by the INACT_ODR[1:0] bits in CTRL4 although the content of CTRL5 is left untouched. The duration of the inactivity status to be recognized is defined by the SLEEP_DUR[3:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to a duration of 512/ODR, where ODR is the accelerometer output data rate configured by the ODR[3:0] bits of the CTRL5 register. The default value is 0000 and corresponds to a duration of 16/ODR.

During the inactivity status of the device, the SLEEP_STATE bit in the WAKE_UP_SRC register is set high. Every time the device status changes from activity to inactivity or vice versa, the SLEEP_CHANGE_IA bit in WAKE_UP_SRC is set for about 1/ODR_ACT + 17 ms if the device status changes from activity to inactivity (for an ODR less than 50 Hz, 37 ms) and for about 17 ms if the device status changes from inactivity to activity. This bit can be routed to the INT1 (or INT2) pin alternatively:

- If the SLEEP_STATUS_ON_INT bit in INTERRUPT_CFG (17h) is set to 1, then the SLEEP_STATE bit in WAKE_UP_SRC is routed to the pin.
- If the SLEEP_STATUS_ON_INT bit in INTERRUPT_CFG (17h) is set to 0, then the SLEEP_CHANGE_IA bit in WAKE_UP_SRC is routed to the pin.

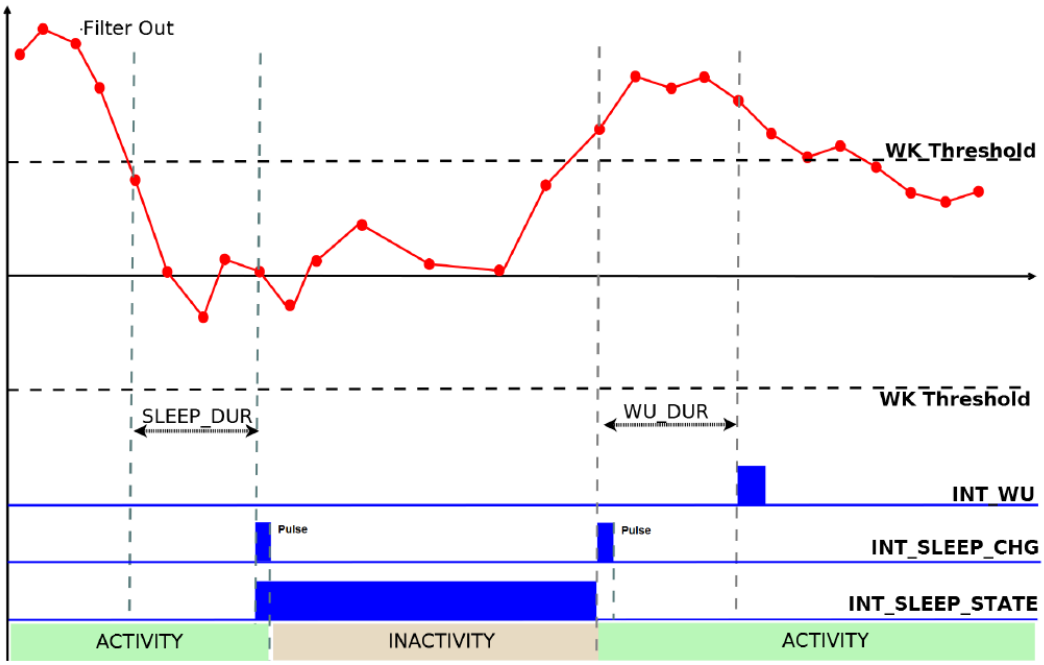
In any case, the INT1_SLEEP_CHANGE (or INT2_SLEEP_CHANGE) bit of the MD1_CFG (or MD2_CFG) register has to be set to 1. Note that the interrupt mode of these signals is not configurable like the other interrupt signals.

When in inactivity state, in order to change the value of the CTRL5 register (and eventually to change the operating mode), it is necessary to disable the INTERRUPTS_ENABLE bit of the INTERRUPT_CFG register first. If the INTERRUPTS_ENABLE bit needs to remain enabled, it is necessary to follow this procedure to change the value of the CTRL5 register: disable the INTERRUPTS_ENABLE bit, switch to power-down mode, re-enable the INTERRUPTS_ENABLE bit, and finally change the value of the CTRL5 register as desired.

When a single sample of slope filtered data on one axis becomes bigger than the threshold, the CTRL5 register settings are immediately restored (activity). The wake-up interrupt event can be delayed in function of the value of the WU_DUR[1:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to a duration of 1/ODR, where ODR is the accelerometer output data rate. In order to generate the interrupt at the same time as the inactivity/activity event, WU_DUR[1:0] have to be set to 0.

When the wake-up event is detected, the interrupt is set high for a duration of 1/ODR, then it is automatically deasserted (the WU_IA event on the pin must be routed by setting the INT1_WU (or INT2_WU) bit of the MD1_CFG (or MD2_CFG) register to 1).

Figure 15. **Activity/inactivity recognition**



The following routine describes the implementation of activity/inactivity detection with routing the wake-up event to the INT1 pin.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write C0h in CTRL4 // Select ODR during inactivity status
3. Write 17h in CTRL1 // Enable wake-up event detection on the X, Y, and Z axes
4. Write 42h in WAKE_UP_DUR // Set duration for inactivity detection
   // Set duration for wake-up detection
5. Write 42h in WAKE_UP_THS // Set activity/inactivity threshold
   // Enable activity/inactivity detection
6. Write 20h in MD1_CFG // Activity (wake-up) interrupt driven to INT1 pin
7. Write 01h in INTERRUPT_CFG // Enable basic interrupts
8. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

In this example the WU_THS field of the WAKE_UP_THS register is set to 000010, therefore the activity/ inactivity threshold is 62.5 mg (= 2 * FS / 64).

Before inactivity detection, the X,Y,Z slope filtered 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 0010, corresponding to 5.12 s (= 2 * 512 / ODR). After this period of time has elapsed, the accelerometer ODR is internally set to 25 Hz in ultralow-power mode since the INACT_ODR field of the CTRL4 register is set to 11.

The activity status is detected and the CTRL5 register settings immediately restored if the slope filtered data of (at least) one axis is bigger than the threshold and the wake-up interrupt was notified after an interval defined by the WU_DUR field of the WAKE_UP_DUR register: this field is set to 10, corresponding to 10 ms (= 2 * 1 / ODR).

The following routine describes how to route the sleep change event to the INT1 pin.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write C0h in CTRL4 // Select ODR during inactivity status
3. Write 42h in WAKE_UP_DUR // Set duration for inactivity detection
4. Write 42h in WAKE_UP_THS // Set activity/inactivity threshold
   // Enable activity/inactivity detection
5. Write 80h in MD1_CFG // Sleep change interrupt driven to INT1 pin
6. Write 01h in INTERRUPT_CFG // Enable basic interrupts
7. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

The following routine describes how to route the sleep status event to the INT1 pin.

1. Write 00h in CTRL5 // Switch to power-down mode
2. Write C0h in CTRL4 // Select ODR during inactivity status
3. Write 42h in WAKE_UP_DUR // Set duration for inactivity detection
4. Write 42h in WAKE_UP_THS // Set activity/inactivity threshold
   // Enable activity/inactivity detection
5. Write 80h in MD1_CFG // Sleep change interrupt driven to INT1 pin
6. Write 09h in INTERRUPT_CFG // Route sleep status instead of sleep change
   // Enable basic interrupts
7. Write 90h in CTRL5 // ODR = 200 Hz, low-power mode, FS ±2 *g*

The last two examples are similar to the first one except for the event routed to the INT1 pin.

5.6.1 Stationary/motion detection

Stationary/motion detection is a particular case of the activity/inactivity functionality in which no ODR / power mode changes occur when a sleep condition (equivalent to stationary condition) is detected. Stationary/motion detection is activated by setting the INACT_ODR[1:0] = 00 bits in the CTRL4 register.

5.7 Boot status

After the power-up command is performed, the LIS2DUXS12 performs a 25 ms boot procedure to load the trimming parameters. After the boot is completed, the accelerometer is automatically configured in soft power-down mode.

After power-up, the trimming parameters can be reloaded by setting the BOOT bit of the CTRL4 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 does not change after boot. During boot time, the device operating mode must not be changed.

If a reset to the default value of the control registers is required too (registers addresses: 0Ch, 0Eh, and from 10h to 20h), it can be performed by setting the SW_RESET bit of the CTRL1 register to 1. The software reset procedure can take up to 50 µs. All the control registers involved in the software reset procedure cannot be accessed for read/write operations during this period.

The boot status signal is driven to the INT1 (or INT2) interrupt pin by setting the INT1_BOOT (or INT2_BOOT) bit of the CTRL2 (or CTRL3) register to 1. The signal goes to 1 while a boot is taking place, and returns to 0 when it is done.

The flow must be performed serially (from ANY operating mode) as shown in the example below:

1. Set the device in power-down mode.
2. Set the SW_RESET bit to 1.
3. Wait 50 µs.
4. Check that the SW_RESET bit of the CTRL1 register has returned to 0.
5. Set the device in the desired operating mode.
6. Set the BOOT bit to 1.
7. Wait 25 ms.
8. Check that the BOOT bit of the CTRL4 register has returned to 0.

Note: The SIM bit of the PIN_CTRL register is not reset during the software reset procedure.

6 Embedded functions

The device implements in hardware many embedded functions; specific IP blocks with negligible power consumption and high-level performance implement the following functions:

- Pedometer functions (step detector and step counter)
- Significant motion
- Relative tilt
- Timestamp

The execution status of the embedded functions can be monitored with the EMB_FUNC_ENDOP bit of the EMB_FUNC_EXEC_STATUS embedded functions register. This bit is set to 1 when no embedded function is running. This register also contains the EMB_FUNC_EXEC_OVR bit, which is set to 1 when the execution of the embedded functions exceeds the maximum time, and new data are generated before the end of the execution of the algorithms.

By setting the SMART_POWER_EN bit of the CTRL1 register, smart power management is enabled. This feature dynamically scales the supply current of the embedded functions to save on power consumption. When this feature is disabled, the supply current of the hardware blocks executing the embedded functions is always kept at the operating level. When this feature is enabled, the supply current of these hardware blocks is kept at the operating level only when necessary (that is, when the embedded functions are running), otherwise it is kept at a lower non-operating level; in this way, their total supply current is reduced.

The duration of the execution of the algorithms of the embedded functions is internally measured and compared with a programmable duration threshold. If the duration of the execution of the algorithms is lower than this threshold, the supply current of the hardware blocks is lowered as soon as the embedded functions stop running, and it is brought back to the operating level when the embedded functions resume execution. Otherwise, if the duration of the execution of the algorithms is greater than this threshold, the supply current of the hardware blocks is always kept at the operating level. The duration threshold can be configured through the SMART_POWER_CTRL_DUR[3:0] bits of the SMART_POWER_CTRL embedded advanced feature register. The user can get an idea of the value of the duration threshold to configure by analyzing the execution time of the algorithms of the embedded functions through the EMB_FUNC_ENDOP bit.

The comparison between the duration of the execution of the algorithms and the duration threshold can be performed at each data-ready window, or it can be performed during a programmable number of consecutive windows after which a definitive decision on the enabling of the smart power management feature is taken (and the comparison is not performed anymore). If in at least one data-ready window the duration of the execution of the algorithms is greater than the duration threshold, the feature is disabled to guarantee that the execution of the algorithms is always completed. The number of windows in which the comparison is performed can be configured through the SMART_POWER_CTRL_WIN[3:0] bits of the SMART_POWER_CTRL embedded advanced feature register. If these bits are set to 0000, the comparison is performed at each data-ready window.

6.1 Pedometer functions: step detector and step counter

A specific IP block is dedicated to pedometer functions: the step detector and the step counter.

Pedometer functions work at 25 Hz; consequently the accelerometer ODR must be set at a value of 25 Hz or higher when using them.

In order to enable the pedometer functions, it is necessary to set the PEDO_EN bit of the EMB_FUNC_EN_A embedded functions register to 1. The algorithm internal state can be reinitialized by asserting the STEP_DET_INIT bit of the EMB_FUNC_INIT_A embedded functions register.

The step counter indicates the number of steps detected by the step detector algorithm after the pedometer function has been enabled. The step count is given by the concatenation of the STEP_COUNTER_H and STEP_COUNTER_L embedded functions registers and it is represented as a 16-bit unsigned number.

The step count is not reset to zero when the accelerometer is configured in power-down or the pedometer is disabled or reinitialized; it can be reset to zero by setting the PEDO_RST_STEP bit of the EMB_FUNC_SRC register to 1. After the counter resets, the PEDO_RST_STEP bit is automatically set back to 0.

The step detector functionality generates an interrupt every time a step is recognized. In the case of interspersed step sessions, 10 consecutive steps (debounce steps) have to be detected before the first interrupt generation in order to avoid false step detections (debounce functionality).

The number of debounce steps can be modified through the DEB_STEP[7:0] bits of the PEDO_DEB_STEPS_CONF register in the embedded advanced features registers. Basically, it corresponds to the minimum number of steps to be detected before the first step counter increment. 1 LSB of this field corresponds to 1 step, the default value is 10 steps. The debounce functionality restarts after around 1.2 s of device inactivity.

An additional block (the false-positive rejection block) can perform real-time recognition of specific conditions and therefore adapt the pedometer operation. This feature can be enabled by setting the MLC_EN bit of the EMB_FUNC_EN_B register and the FP_REJECTION_EN bit of the PEDO_CMD_REG embedded advanced features register to 1. This block performs real-time recognition of walking activity based on statistical data and inhibits the step counter if no walking activity is detected.

STMicroelectronics provides the tools to generate specific pedometer configurations starting from a set of data-logs with a reference number of steps.

The EMB_FUNC_SRC embedded functions register contains some read-only bits related to the pedometer function state.

Table 22. EMB_FUNC_SRC embedded functions register

b7	b6	b5	b4	b3	b2	b1	b0
PEDO_RST _STEP	0	STEP_ DETECTED	STEP_COUNT _DELTA_IA	STEP_ OVERFLOW	STEPCOUNTER _BIT_SET	0	0

- PEDO_RST_STEP: pedometer step counter reset. It can be set to 1 to reset the number of steps counted. It is automatically set back to 0 after the counter reset.
- STEP_DETECTED: step detector event status. It signals a step detection (after the debounce).
- STEP_COUNT_DELTA_IA: instead of generating an interrupt signal every time a step is recognized, it is possible to generate it if at least one step is detected within a certain time period, defined by setting a value different from 00h in the PEDO_SC_DELTAT_H and PEDO_SC_DELTAT_L embedded advanced features (page 1) registers. It is necessary to set the TIMESTAMP_EN bit of the INTERRUPT_CFG register to 1 (to enable the timer). The time period is given by the concatenation of PEDO_SC_DELTAT_H and PEDO_SC_DELTAT_L and it is represented as a 16-bit unsigned value with a resolution of 2.56 ms. STEP_COUNT_DELTA_IA goes high (at the end of each time period) if at least one step is counted (after the debounce) within the programmed time period. If the time period is not programmed (PEDO_SC_DELTAT = 0), this bit is kept at 0. Since timestamp information is not accurate in ultralow-power mode, this interrupt signal should not be used in this mode.
- STEP_OVERFLOW: overflow signal, which goes high when the step counter value reaches 2^{16}.
- STEPCOUNTER_BIT_SET: step counter event status. It signals an increase in the step counter (after the debounce). If a timer period is programmed in the PEDO_SC_DELTAT_H and PEDO_SC_DELTAT_L embedded advanced features registers, this bit is kept at 0.

The step detection interrupt signal can also be checked by reading the IS_STEP_DET bit of the EMB_FUNC_STATUS embedded functions register or the IS_STEP_DET bit of the EMB_FUNC_STATUS_MAINPAGE register.

The IS_STEP_DET bit can have different behaviors, as summarized in the table below, depending on the value of the PEDO_SC_DELTAT bit in the EMB_FUNC_SRC embedded functions register and the CARRY_COUNT_EN bit in the PEDO_CMD_REG embedded advanced features register.

Table 23. IS_STEP_DET configuration

PEDO_SC_DELTAT	CARRY_COUNT_EN	IS_STEP_DET
PEDO_SC_DELTAT = 0	0	STEPCOUNTER_BIT_SET
PEDO_SC_DELTAT > 0	0	STEP_COUNT_DELTA_IA
PEDO_SC_DELTAT ≥ 0	1	STEP_OVERFLOW

The IS_STEP_DET interrupt signal can be driven to the INT1/INT2 interrupt pin by setting the INT1_STEP_DETECTOR/INT2_STEP_DETECTOR bit of the EMB_FUNC_INT1/EMB_FUNC_INT2 register to 1. In this case it is mandatory to also enable routing the embedded functions event to the INT1/INT2 interrupt pin by setting the INT1_EMB_FUNC/INT2_EMB_FUNC bit of the MD1_CFG/MD2_CFG register.

The behavior of the interrupt signal is pulsed by default. The duration of the pulse is equal to 1/25 Hz. Latched mode can be enabled by setting the EMB_FUNC_LIR bit of the PAGE_RW embedded functions register to 1. In this case, the interrupt signal is reset by reading the IS_STEP_DET bit of the EMB_FUNC_STATUS embedded functions register or the IS_STEP_DET bit of the EMB_FUNC_STATUS_MAINPAGE register.

The step counter can be batched in FIFO (see Section 7: First-in first-out (FIFO) buffer for details).

A basic software routine that shows how to enable step counter detection is as follows:

1.	Write 10h to CTRL4	// Enable embedded functions
2.	Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 1	// Enable access to embedded functions registers
2.	Write 40h to PAGE_RW	// Select write operation mode
4.	Write 01h to PAGE_SEL	// Select page 0
5.	Write 5Dh to PAGE_ADDR	// Set embedded advanced features register to be written (PEDO_CMD_REG)
6.	Write 04h to PAGE_VALUE	// Enable false-positive rejection block (FP_REJECTION_EN = 1)
7.	Write 00h to PAGE_RW	// Write operation mode disabled
8.	Write 08h to EMB_FUNC_EN_A	// Enable pedometer
9.	Write 10h to EMB_FUNC_EN_B	// Enable pedometer false-positive rejection block (MLC_EN = 1)
10.	Write 08h to EMB_FUNC_INT1	// Step detection interrupt driven to INT1 pin
11.	Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 0	// Disable access to embedded functions registers
12.	Write 01h to MD1_CFG	// Enable routing the embedded functions interrupt
13.	Write 61h to CTRL5	// Turn on the accelerometer (ODR = 25 Hz, FS = ±4 *g*)

6.2 Significant motion

The significant motion function generates an interrupt when a ‘significant motion’, that could be due to a change in user location, is detected. In the device, this function has been implemented in hardware using only the accelerometer.

The significant motion functionality can be used in location-based applications in order to receive a notification indicating when the user is changing location.

The significant motion function works at 25 Hz, so the accelerometer ODR must be set at a value of 25 Hz or higher. It generates an interrupt when the difference between the number of steps counted from its initialization/ reset is higher than 10 steps. After an interrupt generation, the algorithm internal state is reset.

In order to enable significant motion detection, it is necessary to set the SIGN_MOTION_EN bit of the EMB_FUNC_EN_A embedded functions register to 1. The algorithm can be reinitialized by asserting the SIG_MOT_INIT bit of the EMB_FUNC_INIT_A embedded functions register.

Note: The significant motion feature automatically enables the internal step counter algorithm.

The significant motion interrupt signal can be driven to the INT1/INT2 interrupt pin by setting the INT1_SIG_MOT/ INT2_SIG_MOT bit of the EMB_FUNC_INT1/EMB_FUNC_INT2 register to 1. In this case, it is mandatory to also enable the embedded functions event routing to the INT1/INT2 interrupt pin by setting the INT1_EMB_FUNC/ INT2_EMB_FUNC bit of the MD1_CFG/MD2_CFG register.

The significant motion interrupt signal can also be checked by reading the IS_SIGMOT bit of the EMB_FUNC_STATUS embedded functions register or the IS_SIGMOT bit of the EMB_FUNC_STATUS_MAINPAGE register.

The behavior of the significant motion interrupt signal is pulsed by default. The duration of the pulse is equal to 1/25 Hz. Latched mode can be enabled by setting the EMB_FUNC_LIR bit of the PAGE_RW embedded functions register to 1: in this case, the interrupt signal is reset by reading the IS_SIGMOT bit of the EMB_FUNC_STATUS embedded functions register or the IS_SIGMOT bit of the EMB_FUNC_STATUS_MAINPAGE register.

A basic software routine which shows how to enable significant motion detection is as follows:

1. Write 10h to CTRL4 // Enable embedded functions
2. Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 1 // Enable access to embedded functions registers
3. Write 20h to EMB_FUNC_EN_A // Enable significant motion detection
4. Write 20h to EMB_FUNC_INT1 // Significant motion interrupt driven to INT1 pin
5. Write 80h to PAGE_RW // Enable latched mode for embedded functions
6. Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 0 // Disable access to embedded functions registers
7. Write 01h to MD1_CFG // Enable routing the embedded functions interrupt
8. Write 60h to CTRL5 // Turn on the accelerometer (ODR = 25 Hz, FS = ±2 *g*)

6.3 Relative tilt

The tilt function allows detecting when an activity change occurs (for example, when a phone is in a front pocket and the user goes from sitting to standing or from standing to sitting). In the device it has been implemented in hardware using only the accelerometer.

The tilt function works at 25 Hz, so the accelerometer ODR must be set at a value of 25 Hz or higher.

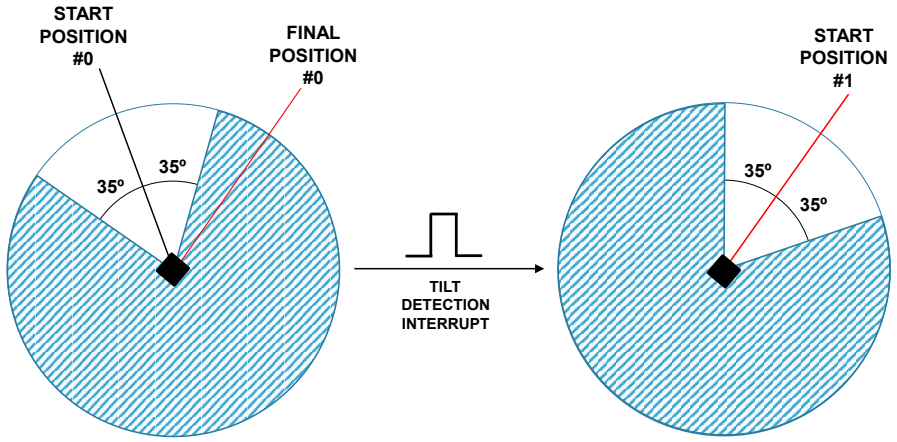
In order to enable the relative tilt detection function, it is necessary to set the TILT_EN bit of the EMB_FUNC_EN_A embedded functions register to 1. The algorithm can be reinitialized by asserting the TILT_INIT bit of the EMB_FUNC_INIT_A embedded functions register.

If the device is configured for tilt event detection, an interrupt is generated when the device is tilted by an angle greater than 35 degrees from the start position. The start position is defined as the position of the device when the tilt detection is enabled/re-initialized or the position of the device when the last tilt interrupt was generated.

After this function is enabled or reinitialized, the tilt logic typically requires a 2-second settling time before being able to generate the first interrupt.

In the example shown in Figure 16 tilt detection is enabled when the device orientation corresponds to “start position #0”. The first interrupt is generated if the device is rotated by an angle greater than 35 degrees from the start position. After the first tilt detection interrupt is generated, the new start position (#1) corresponds to the position of the device when the previous interrupt was generated (final position #0), and the next interrupt signal is generated as soon as the device is tilted by an angle greater than 35 degrees, entering the blue zone surrounding the start position #1.

Figure 16. Tilt example



The tilt interrupt signal can be driven to the INT1/INT2 interrupt pin by setting the INT1_TILT/INT2_TILT bit of the EMB_FUNC_INT1/EMB_FUNC_INT2 register to 1. In this case it is mandatory to also enable the embedded functions event routing to the INT1/INT2 interrupt pin by setting the INT1_EMB_FUNC/INT2_EMB_FUNC bit of MD1_CFG/MD2_CFG register.

The tilt interrupt signal can also be checked by reading the IS_TILT bit of the EMB_FUNC_STATUS embedded functions register or the IS_TILT bit of the EMB_FUNC_STATUS_MAINPAGE register.

The behavior of the tilt interrupt signal is pulsed by default. The duration of the pulse is equal to 1/25 Hz. Latched mode can be enabled by setting the EMB_FUNC_LIR bit of the PAGE_RW embedded functions register to 1. In this case, the interrupt signal is reset by reading the IS_TILT bit of the EMB_FUNC_STATUS embedded functions register or the IS_TILT bit of the EMB_FUNC_STATUS_MAINPAGE register.

Hereafter a basic software routine which shows how to enable the tilt detection function:

1. Write 10h to CTRL4	// Enable embedded functions
2. Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 1	// Enable access to embedded functions registers
3. Write 10h to EMB_FUNC_EN_A	// Enable tilt detection
4. Write 10h to EMB_FUNC_INT1	// Tilt interrupt driven to INT1 pin
5. Write 80h to PAGE_RW	// Enable latched mode for embedded functions
6. Set EMB_FUNC_REG_ACCESS bit of FUNC_CFG_ACCESS to 0	// Disable access to embedded functions registers
7. Write 01h to MD1_CFG	// Enable embedded functions interrupt routing
8. Write 60h to CTRL5	// Turn on the accelerometer (ODR = 25 Hz, FS = ±2 *g*)

6.4 Timestamp

Together with sensor data the device can provide timestamp information with a resolution of 10 µs.

To enable this functionality the TIMESTAMP_EN bit of the INTERRUPT_CFG 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.

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 by writing the reset value AAh in the TIMESTAMP2 register. The reset procedure must be performed only after the accelerometer has been turned on. After the reset value has been written into the register, at least 350 µs must elapse before performing another write transaction.

The bit INT1_TIMESTAMP/INT2_TIMESTAMP enables routing the alert of the timestamp registers overflow (happening within 2.5 ms) to the INT1/INT2 pin.

The timestamp can be batched in FIFO (see Section 7: First-in first-out (FIFO) buffer for details). Timestamp information is not accurate in ultralow-power mode and should not be relied upon in this mode.

7 First-in first-out (FIFO) buffer

In order to limit intervention by the host processor and facilitate post-processing data for event recognition, the LIS2DUXS12 embeds a first-in, first-out buffer (FIFO).

The FIFO can be configured to store the following data:

- Accelerometer sensor data
- Analog hub / Qvar sensor data
- Timestamp data
- Configuration-change data
- Temperature sensor data
- Step counter data
- MLC results (features, filters, and outputs)
- FSM outputs and long counter

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 41h to address 46h: 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 TAG_SENSOR_[4:0] field of the FIFO_DATA_OUT_TAG (40h) 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_CTRL register:

- Bypass mode
- FIFO mode
- Continuous mode
- Continuous-to-FIFO mode
- Bypass-to-continuous mode
- Bypass-to-FIFO mode

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

A programmable FIFO threshold can be set in the FIFO_WTM register using the FTH[6:0] 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 CTRL2 register, and through the INT2_ FIFO_FULL, INT2_FIFO_FTH and INT2_FIFO_OVR bits of the CTRL3 register.

7.1 FIFO description and batched sensors

FIFO is divided into 128 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 896 bytes and can contain 768 bytes of data. The TAG byte contains the information indicating which data is stored in the FIFO data field and other useful information (see Section 7.2.7: FIFO_DATA_OUT_TAG).

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 sensors, which are physical sensors:
   a. Accelerometer sensor
   b. Analog hub / Qvar
   c. Temperature sensor
2. Auxiliary sensors, which contain information of the status of the device:
   a. Timestamp sensor
   b. Configuration-change sensor (CFG-change)
3. Virtual sensors:
   a. Machine learning core filters, features, and outputs
   b. Finite state machine outputs and long counter
   c. Step counter sensor

A write to FIFO can be triggered by events of the main sensors or by events of the virtual sensors.

In order to maximize the number of data collected in FIFO, it is possible to enable 2x depth, writing the FIFO_DEPTH bit in the FIFO_CTRL (15h) register. If 2x depth is enabled, the most significant 8 bits for each accelerometer axis are stored in FIFO: each FIFO word contains data from two consecutive ODRs, the current and the previous one. In this case, the temperature data (or analog hub / Qvar data) is not stored in FIFO but is available in the OUT_T_AH_QVAR_L (2Eh) and OUT_T_AH_QVAR_H (2Fh) registers.

7.2 FIFO registers

The FIFO buffer is managed by:

- FIFO_EN bit of the CTRL4 register to enable batching in FIFO, to be set to 1 before configuring the FIFO buffer
- Four configuration registers: FIFO_CTRL, FIFO_WTM, FIFO_BATCH_DEC and EMB_FUNC_FIFO_EN
- 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_FIFO_FULL, INT1_FIFO_OVR, INT1_FIFO_TH bits of the CTRL2 register and INT2_FIFO_FULL, INT2_FIFO_OVR, INT2_FIFO_TH bits of the CTRL3 register

7.2.1 FIFO_CTRL

Table 24. FIFO_CTRL register

b7	b6	b5	b4	b3	b2	b1	b0
CFG_CHG_EN	FIFO_ DEPTH	0	0	STOP_ON _FTH	FIFO_MODE2	FIFO_MODE1	FIFO_MODE0

The FIFO_CTRL register contains the:

- CFG_CHG_EN bit, which enables batching in FIFO the device configuration and timestamp value when the output data rate or the batch data rate changes
- FIFO_DEPTH bit, which enables 2x depth mode for the FIFO buffer: when this mode is enabled, the most significant 8 bits for each acceleration data are stored in FIFO at a rate equal to ODR/2, and each FIFO data word contains data of two consecutive ODRs, the current and the previous one, in the format specified in Table 34
- STOP_ON_FTH bit, which allows limiting the FIFO depth to the watermark threshold level

Moreover, it contains the bits to select the different FIFO modes (FIFO_MODE[2:0]), which are described in Section 7.7: FIFO modes.

7.2.2 FIFO_WTM

Table 25. FIFO_WTM register

b7	b6	b5	b4	b3	b2	b1	b0
XL_ONLY _FIFO	FTH6	FTH5	FTH4	FTH3	FTH2	FTH1	FTH0

The FIFO_WTM register contains the 7 bits to define the FIFO watermark threshold, FTH[6:0], whose maximum value is 127. The FIFO watermark flag (FIFO_WTM_IA bit in the FIFO_STATUS1 register) rises when the number of bytes stored in the FIFO is equal to or higher than the watermark threshold level.

Moreover, this register contains the XL_ONLY_FIFO bit. When 2x depth mode is disabled, it is possible to avoid storing the temperature data (or analog hub / Qvar data) in FIFO by setting this bit to 1. In this case, the accelerometer data are stored in FIFO in 16-bit format at the ODR rate in the format specified in Table 34.

7.2.3 FIFO_BATCH_DEC

Table 26. **FIFO_BATCH_DEC register**

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	DEC_TS_ BATCH_1	DEC_TS_ BATCH_0	BDR_XL_2	BDR_XL_1	BDR_XL_0

The FIFO_WTM register contains the DEC_TS_BATCH_[1:0] bits, which select the decimation factor for timestamp batching in FIFO. The write rate is the accelerometer BDR divided by the decimation factor, as shown in the table below.

Table 27. **Decimation factor for timestamp batching**

DEC_TS_BATCH_[1:0]	Timestamp batch data rate [Hz]
00	Timestamp not batched in FIFO
01	BDR_XL
10	BDR_XL/8
11	BDR_XL/32

The accelerometer batch data rate itself can be selected with the BDR_XL_[2:0] bits of this register, as shown in the table below.

Table 28. **Accelerometer batch data rate**

BDR_XL_[2:0]	Accelerometer batch frequency
000	ODR (default)
001	ODR/2
010	ODR/4
011	ODR/8
100	ODR/16
101	ODR/32
110	ODR/64
111	Accelerometer not batched in FIFO

7.2.4 EMB_FUNC_FIFO_EN

Table 29. EMB_FUNC_FIFO_EN register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	FSM_FIFO_EN	MLC_FILTER_FEATURE_FIFO_EN	MLC_FIFO_EN	STEP_COUNTER_FIFO_EN

This register can be used to enable batching the results from the embedded functions in the FIFO buffer:

- FSM_FIFO_EN enables batching the finite state machine results in the FIFO buffer.
- MLC_FILTER_FEATURER_FIFO_EN enables batching the machine learning core filters and features in FIFO.
- MLC_FIFO_EN enables batching the machine learning core results in the FIFO buffer.
- STEP_COUNTER_FIFO_EN enables batching the step counter values in the FIFO buffer.

When batching any of the results from the embedded functions in the FIFO buffer, the FIFO must be enabled in continuous mode (refer to Section 7.7: FIFO modes).

7.2.5 FIFO_STATUS1

Table 30. FIFO_STATUS1 register

b7	b6	b5	b4	b3	b2	b1	b0
FIFO_WTM_IA	FIFO_OVR_IA	-	-	-	-	-	-

The FIFO_STATUS1 register provides information about the current status of the FIFO buffer. In particular:

- FIFO_WTM_IA is the FIFO watermark status. This bit goes high when the number of FIFO words 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 CTRL2 register or the INT2_FIFO_TH bit of the CTRL3 register.
- FIFO_OVR_IA is the FIFO overrun status. This bit goes high when the FIFO is 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 CTRL2 register or the INT2_FIFO_OVR bit of the CTRL3 register.

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

7.2.6 FIFO_STATUS2

Table 31. FIFO_STATUS2 register

b7	b6	b5	b4	b3	b2	b1	b0
FSS7	FSS6	FSS5	FSS4	FSS3	FSS2	FSS1	FSS0

The FIFO_STATUS2 register provides information about the number of samples stored in the FIFO. 1 LSB of the FSS[7:0] value is referred to as a FIFO word.

When FSS[7:0] is equal to 00000000, the FIFO is empty. When FSS[7:0] is equal to 10000000, FIFO is full and the unread samples are 128. This means that the FSS7 bit is the flag that indicates if the FIFO has been filled. This flag can be driven to the two interrupt pins by setting to 1 the INT1_FIFO_FULL bit of the CTRL2 register or the INT2_FIFO_FULL bit of the CTRL3 register.

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.

7.2.7 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.

Table 32. FIFO_DATA_OUT_TAG register

b7	b6	b5	b4	b3	b2	b1	b0
TAG_ SENSOR_4	TAG_ SENSOR_3	TAG_ SENSOR_2	TAG_ SENSOR_1	TAG_ SENSOR_0	0	0	-

The TAG_SENSOR_[4:0] field identifies the sensors stored in the 6 data bytes. The table below contains all the possible values and associated type of sensor for the bitfield.

Table 33. Identification of sensor in FIFO

TAG_SENSOR_[4:0]	Data in FIFO
00000	FIFO empty(1)
00010	Accelerometer and temperature (XL_ONLY_FIFO = 0) or accelerometer only data (XL_ONLY_FIFO = 1)
00011	Accelerometer only data (2x depth mode) when the analog hub / Qvar is disabled
00100	Timestamp or CFG_CHG
10010	Step counter
11010	MLC result
11011	MLC filter
11100	MLC feature
11101	FSM result
11110	Accelerometer-only data (2x depth mode) when the analog hub / Qvar is enabled
11111	Accelerometer and analog hub / Qvar (XL_ONLY_FIFO = 0) or accelerometer-only data (XL_ONLY_FIFO = 1)

1. *If samples are continuously read before a new sample arrives, the FIFO tag value will continue to be equal to 00000.*

7.2.8 FIFO_DATA_OUT

Data can be retrieved from the FIFO through six dedicated registers, from address 41h to address 46h: 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.

7.3 FIFO batched sensors

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

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

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

7.4 Main sensors

The main sensors are the physical sensors of the LIS2DUXS12 device: the accelerometer, the analog hub / Qvar, and the temperature sensor. The batch data rate can be configured through the BDR_XL_[3:0] field of the FIFO_BATCH_DEC register.

If 2x depth is enabled, the most significant 8 bits for each acceleration data are stored in FIFO, and each FIFO data word contains data of two consecutive ODRs, the current and the previous one. In this case, the analog hub / Qvar data or the temperature sensor data is not stored in FIFO but is available in the OUT_T_AH_QVAR_L (2Eh) and OUT_T_AH_QVAR_H (2Fh) registers. The availability of data depends on whether the analog hub / Qvar is enabled or not. See Section 9: Analog hub and Qvar sensor in order to enable/disable the functionality of the analog hub / Qvar.

If 2x depth is not enabled and the analog hub / Qvar is enabled, the analog hub / Qvar output data is saved as a 12-bit value together with the accelerometer output data (which is saved as a 12-bit value for each axis). If the analog hub / Qvar is disabled, the temperature sensor data is saved instead of the analog hub / Qvar sensor data. This is true only if the XL_ONLY_FIFO bit in the FIFO_WTM (16h) register is set to 0. If the XL_ONLY_FIFO bit is set to 1, only the accelerometer data are stored in FIFO (in 16-bit format at the ODR rate).

The output data format for each case is indicated in the following table.

Table 34. Output data format in FIFO of the accelerometer and Qvar sensors

Data (if FIFO_DEPTH = 1)	Data (if FIFO_DEPTH = 0 and XL_ONLY_FIFO = 0)	Data (if FIFO_DEPTH = 0 and XL_ONLY_FIFO = 1)	FIFO_DATA_OUT registers
ACC_X_PREV[7:0]	ACC_X[7:0]	ACC_X[7:0]	FIFO_DATA_OUT_X_L
ACC_Y_PREV[7:0]	ACC_Y[3:0] and ACC_X[11:8]	ACC_X[15:8]	FIFO_DATA_OUT_X_H
ACC_Z_PREV[7:0]	ACC_Y[11:4]	ACC_Y[7:0]	FIFO_DATA_OUT_Y_L
ACC_X_CURR[7:0]	ACC_Z[7:0]	ACC_Y[15:8]	FIFO_DATA_OUT_Y_H
ACC_Y_CURR[7:0]	QVAR[3:0] and ACC_Z[11:8]	ACC_Z[7:0]	FIFO_DATA_OUT_Z_L
ACC_Z_CURR[7:0]	QVAR[11:4]	ACC_Z[15:8]	FIFO_DATA_OUT_Z_H

7.5 Auxiliary sensors

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

- Timestamp sensor: it stores the timestamp corresponding to a FIFO time slot (the TIMESTAMP_EN bit of the INTERRUPT_CFG register must be set to 1 and the DEC_TS_BATCH_[1:0] bits of the FIFO_BATCH_DEC register must be configured properly).
- CFG-Change sensor: it identifies a change in some configurations of the device (the CFG_CHG_EN bit of the FIFO_CTRL register must be set to 1).

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

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

Table 35. Timestamp output data format in FIFO

Data	FIFO_DATA_OUT registers
TIMESTAMP[7:0]	FIFO_DATA_OUT_Y_L
TIMESTAMP[15:8]	FIFO_DATA_OUT_Y_H
TIMESTAMP[23:16]	FIFO_DATA_OUT_Z_L
TIMESTAMP[31:24]	FIFO_DATA_OUT_Z_H

CFG-Change identifies a runtime change in some of the configurations of the main sensors. The configurations whose changes can be detected by this sensor are the ones included in the CTRL5 register (ODR[3:0], BDR[1:0] and FS[1:0]) and in the FIFO_BATCH_DEC register (DEC_TS_BATCH[1:0] and BDR_XL_[2:0]). When a supported runtime change is applied, this sensor is written at the first new main sensor or virtual sensor event together with timestamp information. However, data related to the first event after a configuration change may not yet be updated to the new configuration.

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 (that is, switching mode, output data rate, and so forth).

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

Table 36. CFG-change output data format in FIFO

Data	FIFO_DATA_OUT registers
HP_EN	FIFO_DATA_OUT_X_H[0]
BW[1:0]	FIFO_DATA_OUT_X_H[2:1]
ODR[3:0]	FIFO_DATA_OUT_X_H[6:3]
Configuration change(1)	FIFO_DATA_OUT_X_H[7]
BDR_XL_[2:0]	FIFO_DATA_OUT_Y_H[2:0]
DEC_TS_BATCH[1:0]	FIFO_DATA_OUT_Y_H[4:3]
FS[1:0]	FIFO_DATA_OUT_Y_H[6:5]
AH_QVAR_EN	FIFO_DATA_OUT_Y_H[7]

1. *The TAG used for the TIMESTAMP and CFG_CHANGE sensors is the same, so this bit is used to distinguish what has triggered the write in FIFO. If this bit is 0, no configuration change has been performed. If it is 1, a configuration change has occurred.*

7.6 Virtual sensors

Virtual sensors are divided into the following categories:

1. MLC-generated sensors
2. FSM-generated sensors
3. Step counter sensor

7.6.1 MLC-generated sensors

The following machine learning core (MLC-generated) virtual sensors can be stored in FIFO:

- Results
- Filters and other recursive features
- Windowed features

In order to store MLC-generated sensors in FIFO, the MLC block must be enabled by setting either the MLC_BEFORE_FSM_EN bit of the EMB_FUNC_EN_A register or the MLC_EN bit of the EMB_FUNC_EN_B register.

Batching MLC results is enabled by setting the MLC_FIFO_EN bit of the EMB_FUNC_FIFO_EN register to 1.

An MLC result contains the information of the corresponding MLCx_SRC register, and it is stored in FIFO when a change in the corresponding MLCx_SRC occurs.

Batching MLC filters or features is selectively enabled using one of the tools for configuring the MLC provided by STMicroelectronics. In addition, the MLC_FILTER_FEATURE_FIFO_EN bit of the EMB_FUNC_FIFO_EN register must be set to 1 to globally enable storing MLC filters or features in FIFO.

MLC recursive features (including MLC filters) are stored in FIFO at a rate equivalent to the MLC output data rate (MLC_ODR[2:0] bits). If the filter is applied to the X, Y, Z axes of the desired sensor, one word is stored in FIFO for every axis. If the filter is applied to the norm of the desired sensor, one word is stored in FIFO.

MLC-windowed features are stored in FIFO at the end of every window.

The format of MLC results, features, and filters in FIFO is indicated in the following tables.

Table 37. MLC results in FIFO

Data	FIFO_DATA_OUT registers
MLCx_SRC	FIFO_DATA_OUT_X_L
Index of MLC_SRC	FIFO_DATA_OUT_X_H
TIMESTAMP[7:0]	FIFO_DATA_OUT_Y_L
TIMESTAMP[15:8]	FIFO_DATA_OUT_Y_H
TIMESTAMP[23:16]	FIFO_DATA_OUT_Z_L
TIMESTAMP[31:24]	FIFO_DATA_OUT_Z_H

Table 38. MLC filters or features in FIFO

Data	FIFO_DATA_OUT registers
VALUE[7:0](1)	FIFO_DATA_OUT_X_L
VALUE[15:8](1)	FIFO_DATA_OUT_X_H
IDENTIFIER[7:0](2)	FIFO_DATA_OUT_Y_L
IDENTIFIER[15:8](2)	FIFO_DATA_OUT_Y_H
Reserved	FIFO_DATA_OUT_Z_L
Reserved	FIFO_DATA_OUT_Z_H

1. *This value is represented as a half-precision floating-point number.*
2. *Filter and feature identifiers are indicated in the configuration file generated by STMicroelectronics tools for configuring the MLC.*

7.6.2 FSM-generated sensors

The FSM-generated results can be stored in FIFO as virtual sensors.

In order to store FSM-generated sensors in FIFO, the FSM block must be enabled by setting the FSM_EN bit of the EMB_FUNC_EN_B register. Batching FSM results is enabled by setting the FSM_FIFO_EN bit of the EMB_FUNC_FIFO_EN register to 1.

Table 39. FSM output in FIFO

Data	FIFO_DATA_OUT registers
FSM_OUTx	FIFO_DATA_OUT_X_L
Index of FSM_OUT	FIFO_DATA_OUT_X_H
TIMESTAMP[7:0]	FIFO_DATA_OUT_Y_L
TIMESTAMP[15:8]	FIFO_DATA_OUT_Y_H
TIMESTAMP[23:16]	FIFO_DATA_OUT_Z_L
TIMESTAMP[31:24]	FIFO_DATA_OUT_Z_H

Moreover, when the FSM long counter (saved in registers FSM_LONG_COUNTER_L and FSM_LONG_COUNTER_H) reaches the timeout value (defined in registers FSM_LC_TIMEOUT_L and FSM_LC_TIMEOUT_H), it is stored in FIFO with the format indicated in the following table. This occurs even if the FSM_FIFO_EN bit of the EMB_FUNC_FIFO_EN register is set to 0.

Table 40. FSM long counter in FIFO

Data	FIFO_DATA_OUT registers
FSM_LONG_COUNTER_L	FIFO_DATA_OUT_X_L
00	FIFO_DATA_OUT_X_H
TIMESTAMP[7:0]	FIFO_DATA_OUT_Y_L
TIMESTAMP[15:8]	FIFO_DATA_OUT_Y_H
TIMESTAMP[23:16]	FIFO_DATA_OUT_Z_L
TIMESTAMP[31:24]	FIFO_DATA_OUT_Z_H

7.6.3 Step counter sensor

Step counter data, with associated timestamp, can be stored in FIFO. It is not a continuous rate sensor: the step detection event triggers its writing in FIFO.

In order to enable the step counter sensor in FIFO, the user should:

1. Enable the step counter sensor (set the PEDO_EN bit to 1 in the EMB_FUNC_EN_A embedded functions register).
2. Enable step counter batching (set the STEP_COUNTER_FIFO_EN bit to 1 in the EMB_FUNC_FIFO_EN embedded functions register).

The format of the step counter data read from FIFO is shown in the table below.

Table 41. Step counter output data format in FIFO

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

7.7 FIFO modes

The LIS2DUXS12 FIFO buffer can be configured to operate in six different modes selectable by the FIFO_MODE[2:0] field in the FIFO_CTRL register. Available configurations ensure a high-level of flexibility and extend the number of usable functions in application development.

Bypass, FIFO, continuous, bypass-to-continuous, continuous-to-FIFO and bypass-to-FIFO modes are described in the following sections.

7.7.1 Bypass mode

When bypass mode is enabled, the FIFO is not operational: buffer content is cleared and the FIFO buffer remains empty until another mode is selected.

Bypass mode can be activated by setting the FIFO_MODE[2:0] field to 000 in the FIFO_CTRL register and it is actually enabled 100 µs after setting the FIFO_MODE[2:0] field, if the device is configured in continuous conversion mode. Otherwise, if the device is in one-shot mode, it is recommended to read back the register in order to verify the FIFO bypass mode configuration.

Bypass mode must be used in order to stop and reset the FIFO buffer when a different operating mode or a different FIFO mode is intended to be used. Note that placing the FIFO buffer in bypass mode clears the entire content of the buffer.

7.7.2 FIFO mode

In FIFO mode, the buffer continues filling until full (128 sample sets stored). As soon as the FSS7 bit of the FIFO_STATUS2 register goes to 1, the FIFO stops collecting data and its content remains unchanged until a different mode is selected.

FIFO mode is activated by setting the FIFO_MODE[2:0] field to 001 in the FIFO_CTRL register.

By selecting this mode, FIFO starts data collection and FSS[7:0] changes according to the number of sets of samples stored. At the end of the procedure, the FFS7 bit rises to 1, and data can then be retrieved, performing a 128 sample set read from the FIFO output registers. Communication speed is not so important in FIFO mode because data collection is stopped and there is no risk of overwriting acquired data. Before restarting FIFO mode, at the end of the reading procedure it is necessary to enable bypass mode.

In order to serve the FIFO full event as soon as possible, it is recommended to route it to the INT1/INT2 pin in order to generate an interrupt, which is then be managed by a specific handler:

1. Set the INT1_FIFO_FULL/INT2_FIFO_FULL bit to 1: enables the FIFO full event interrupt.
2. Set FIFO_MODE[2:0] = 001: enables FIFO mode.
3. When the FIFO full event interrupt is generated or the FSS7 bit is high (polling mode), read data from the FIFO_DATA_OUT (from 40h to 46h) output registers.

When FIFO mode is enabled, the buffer starts to collect data and fills all 128 slots (from F0 to F127) at the selected output data rate. When the buffer is full the FSS7 bit goes high and data collection is permanently stopped; the user can decide to read FIFO content at any time because it is maintained unchanged until bypass mode is selected.

The read procedure may be performed inside an interrupt handler triggered by the FIFO full event and it is composed of 128 sample sets of 7 bytes for a total of 896 bytes and retrieves data starting from the oldest sample stored in FIFO (F0). The bypass mode setting resets FIFO and allows the user to enable FIFO mode again.

7.7.3 Continuous mode

In continuous mode, FIFO continues filling, when the buffer is full, the FIFO index restarts from the beginning and older data is replaced by current data. The oldest values continue to be overwritten until a read operation frees FIFO slots. The host processor reading speed is most important in order to free slots faster than new data is made available. Continuous mode can be disabled by enabling bypass mode from the FIFO_MODE[2:0] bits.

Follow these steps for configuring FIFO in continuous mode and for setting a threshold to generate an interrupt on the INT1/INT2 pin in order to trigger a read by the application processor:

1. Set FTH[6:0] to the desired threshold value to trigger the FIFO threshold event.
2. Set the INT1_FIFO_TH/INT2_FIFO_TH bit to 1: enables FIFO threshold event interrupt.
3. Activate continuous mode by setting the FIFO_MODE[2:0] field to 110 in the FIFO_CTRL register.
4. When the FIFO threshold event flag (FIFO_WTM_IA bit in the FIFO_STATUS1 register) goes high and the FIFO threshold event interrupt is generated, read data from the FIFO_DATA_OUT (from 40h to 46h) output registers.

When continuous mode is enabled, the FIFO buffer is continuously filling (from F0 to F127) at the selected output data rate (or double the output data rate if 2x depth mode is selected). When the buffer reaches the desired threshold value, the FIFO threshold event interrupt goes high, and the application processor may read all FIFO samples as soon as possible, avoiding loss of data but at the same time limiting the intervention of the application processor, which increases system efficiency. In case at least one sample is overwritten to store the new data, the FIFO overrun flag (FIFO_OVR_IA bit) goes high. See Section 7.8: Retrieving data from FIFO for more details on FIFO reading speed.

7.7.4 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 the selected interrupt (that is, wake-up, free-fall, single/double/triple-tap, 6D/4D, or any combination of these) occurs and is maintained until bypass mode is set. This mode is sensitive to the level of the interrupt signal, so the interrupt signal must be reset before configuring it.

This mode can be used in order to analyze the history of samples that generated an interrupt; the standard operation is to read FIFO content when a FIFO mode is triggered and the FIFO buffer is full and stopped.

Follow these steps for continuous-to-FIFO mode configuration:

1. Configure the desired interrupt generator by following the instructions in Section 5: Interrupt generation.
2. Activate continuous-to-FIFO mode by setting the FIFO_MODE[2:0] field to 011 in the FIFO_CTRL register.

Note: When the requested event takes place, the FIFO mode change is triggered if and only if the event flag is routed to the INT1 pin.

While in continuous mode the FIFO buffer continues filling. When the requested event takes place, the FIFO mode changes; then, as soon as the buffer becomes full, the FSS7 bit of the FIFO_STATUS2 register is set high and the FIFO stops collecting data.

7.7.5 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 in bypass mode and switches to continuous mode when the selected interrupt (that is, wake-up, free-fall, single/double/triple-tap, 6D/4D, or any combination of these) occurs and is maintained until bypass mode is set. This mode is sensitive to the level of the interrupt signal, so the interrupt signal must be reset before configuring it. Follow these steps for bypass-to-continuous mode configuration:

1. Configure the desired interrupt generator by following the instructions in Section 5: Interrupt generation.
2. Set FTH[6:0] to the desired threshold value to trigger the FIFO threshold event.
3. Set the INT1_FIFO_TH/INT2_FIFO_TH bit to 1: enables FIFO threshold event interrupt.
4. Activate bypass-to-continuous mode by setting the FIFO_MODE[2:0] field to 100 in the FIFO_CTRL register.
5. When the FIFO threshold event interrupt is generated, read data from the FIFO_DATA_OUT (from 40h to 46h) output registers. The sample that generated the trigger is available in FIFO.

When setting the FIFO in bypass-to-continuous mode, the FIFO is initially in bypass mode, so no samples enter in the FIFO buffer. As soon as an event occurs (for example, a wake-up or a free-fall event) the FIFO switches to continuous mode and starts to store the samples at the configured data rate. When the programmed threshold is reached, the FIFO threshold event interrupt goes high, and the application processor may start reading all available FIFO samples as soon as possible to avoid loss of data.

If the FIFO threshold flag (FIFO_WTM_IA bit in the FIFO_STATUS1 register) is set, it goes to 0 as soon as the first FIFO set is read, creating space for new data. Since the FIFO is still in continuous mode, the FIFO eventually reaches the threshold again and the situation repeats.

7.7.6 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.

This mode is sensitive to the level of the interrupt signal, so the interrupt signal must be reset before configuring it. At the first interrupt event, FIFO changes from bypass mode to FIFO mode, which is maintained until bypass mode is set.

Follow these steps for bypass-to-FIFO mode configuration:

1. Configure the desired interrupt generator by following the instructions in Section 5: Interrupt generation.
2. Set the INT1_FIFO_FULL/INT2_FIFO_FULL bit to 1: enables FIFO full event interrupt.
3. Activate bypass-to-FIFO mode by setting the FIFO_MODE[2:0] field to 111 in the FIFO_CTRL register.
4. When the FIFO full event interrupt is generated, read data from the FIFO_DATA_OUT (from 40h to 46h) output registers. The sample that generated the trigger is available in FIFO.

When setting the FIFO in bypass-to-FIFO mode, the FIFO is initially in bypass mode, so no samples enter in the FIFO buffer. As soon as an event occurs (for example, a wake-up or a free-fall event) the FIFO switches to FIFO mode and starts to store the samples at the configured data rate. When the buffer is full, the FIFO full event interrupt goes high, and the application processor must first reset the interrupt pin level (if not already reset), and then it may start reading all FIFO samples (128 sample sets of 7 bytes).

7.8 Retrieving data from 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/MIPI I3C^{SM} 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 FIFO_STATUS1 and FIFO_STATUS2 registers to check how many words are stored in the FIFO.
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 (FSS[7:0] bits of the 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 in multiples of 7 bytes, starting from the FIFO_DATA_OUT_TAG register. The wraparound 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 Temperature sensor

The LIS2DUXS12 is provided with an internal temperature sensor that is suitable for ambient temperature measurement.

If the sensor is in power-down mode, the temperature sensor is off and shows the last value measured.

The DRDY bit in the STATUS register is set high when a new set of data is available, including temperature.

If the analog hub / Qvar is disabled (see Section 9: Analog hub and Qvar sensor), the temperature output is available in the OUT_T_AH_QVAR_L (2Eh) and OUT_T_AH_QVAR_H (2Fh) registers, stored as two's complement data, left-justified in 12-bit mode. Temperature data can also be stored in FIFO (see Section 7: First-in first-out (FIFO) buffer). The temperature sensor acquisition chain (together with the analog hub / Qvar sensor acquisition chain) can be disabled by setting the T_AH_QVAR_DIS bit in the SELF_TEST (32h) register.

See the table below for temperature sensor details.

Table 42. Temperature sensor characteristics

Parameter	Typ.(1)
Data at 25°C	0 LSB
Temperature refresh rate	ODR
Temperature sensor output change vs. temperature	0.045°C/LSB(2)

1. *Typical specifications are not guaranteed.*
2. *12-bit resolution*

8.1 Example of temperature data calculation

Table 43. Content of output data registers vs. temperature 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 (that is, no offset, no gain error, and so forth).

Table 43. Content of output data registers vs. temperature

Temperature values	OUT_T_AH_QVAR_H	OUT_T_AH_QVAR_L
23.92°C	FEh	80h
25.0°C	00h	00h
26.08°C	01h	80h

9 Analog hub and Qvar sensor

The LIS2DUXS12 embeds an analog hub (AH) chain, which provides the capability to connect an external analog input and convert it to a digital signal for processing.

The same analog chain can be used to implement the Qvar (electric charge variation detection) functionality by connecting external electrodes and an electronic signal conditioning circuit. The external electrodes must be connected to pin 12 (INT1) and/or pin 11 (INT2).

Note: When the analog hub / Qvar feature is enabled, the INT1 and INT2 pins are used to connect the external analog lines or the electrodes to sense the Qvar signal, so the INT1/INT2 pins are no longer available for the accelerometer interrupt signals. If a hardware interrupt signal is needed, it is possible to set the INT1_ON_RES bit of register CTRL1 (10h) to 1 in order to route the interrupt signals configured on the INT1 pin to the RES/EXT_CLK pin.

In the LIS2DUXS12, the analog hub / Qvar functionality can be activated by setting the AH_QVAR_EN bit to 1 in the AH_QVAR_CFG (31h) register.

The AH_QVAR_NOTCH_EN bit and the AH_QVAR_NOTCH_CUTOFF bit in the AH_QVAR_CFG (31h) register are used, respectively, to enable/disable the digital notch filter embedded in the analog hub / Qvar reading chain and to set its cutoff frequency (50 Hz or 60 Hz). The filter is compatible with low-power mode and high-performance mode only.

The AH_QVAR_C_ZIN_[1:0] bits can be used to configure the equivalent input impedance of the analog hub and Qvar buffers. The possible values are summarized in the following table.

Table 44. Analog hub and Qvar equivalent input impedance

AH_QVAR_C_ZIN_[1:0]	Equivalent input impedance [MΩ]
00	520
01	175
10	310
11	75

The AH_QVAR_GAIN_[1:0] bits can be used to configure the input-output gain. The possible values are indicated in the following table.

Table 45. Analog hub and Qvar input-output gain

AH_QVAR_GAIN_[1:0]	Input-output gain
00	0.5
01	1
10	2
11	4

Analog hub / Qvar data are available as two's complement data, left-justified in 12-bit mode in the OUT_T_AH_QVAR_L (2Eh) and OUT_T_AH_QVAR_H (2Fh) registers and are generated at the same rate as the one set for the accelerometer sensor through the ODR[3:0] bits in register CTRL5 (14h). If the analog hub / Qvar functionality is enabled, the temperature sensor data are no longer available on these output registers.

The analog hub / Qvar sensor acquisition chain (together with the temperature sensor acquisition chain) can be disabled by setting the T_AH_QVAR_DIS bit in the SELF_TEST (32h) register.

Analog hub / Qvar data can also be stored in FIFO (see Section 7: First-in first-out (FIFO) buffer) and processed by MLC/FSM logic.

The analog hub / Qvar feature is not compatible with ultralow-power mode.

Additional details about Qvar are available in the application note AN5755 on www.st.com.

10 Self-test

Self-test mode allows checking the sensor functionality without moving it, applying an actuation force to the sensor and simulating a definite input acceleration.

The procedures for the positive and negative self-test are as follows.

Positive self-test

1. Set the device in soft power-down and wait 10 ms.
2. Set the FIFO_EN bit in the CTRL4 (13h) register to 1.
3. Set the XL_ONLY_FIFO bit in the FIFO_WTM (16h) register to 1.
4. Set the ST_SIGN_X and ST_SIGN_Y bits in the CTRL3 (12h) register to 1 and the ST_SIGN_Z bit in the WAKE_UP_DUR (1Dh) register to 0.
5. Set the FIFO_CTRL (15h) register to 00h to empty the FIFO.
6. Set the ST[1:0] bits in the SELF_TEST (32h) register to 10.
7. Set ODR = 200 Hz, BW = ODR/2, FS = ±8 *g* from the CTRL5 (14h) register and wait 100 ms.
8. Set the FIFO_CTRL (15h) register to 01h to start filling the FIFO and wait 25 ms so that at least 5 accelerometer data samples are stored in the FIFO.
9. Read the first 5 samples from FIFO whose tag refers to accelerometer-only data, compute the average for each axis, and save the result in OUT1.
10. Set the device in soft power-down and wait 10 ms.
11. Set the FIFO_CTRL (15h) register to 00h to empty the FIFO.
12. Set the ST[1:0] bits in the SELF_TEST (32h) register to 01.
13. Set ODR = 200 Hz, BW = ODR/2, FS = ±8 *g* from the CTRL5 (14h) register and wait 100 ms.
14. Set the FIFO_CTRL (15h) register to 01h to start filling the FIFO and wait 25 ms so that at least 5 accelerometer data samples are stored in the FIFO.
15. Read the first 5 samples from FIFO whose tag refers to accelerometer-only data, compute the average for each axis, and save the result in OUT2.
16. Set the ST[1:0] bits in the SELF_TEST (32h) register to 00.
17. Set the device in soft power-down and wait 10 ms.
18. Self-test deviation is 2 * |OUT2 - OUT1|. Compute the value for each axis and verify that it falls within the range provided in the datasheet.

Negative self-test

1. Set the device in soft power-down and wait 10 ms.
2. Set the FIFO_EN bit in the CTRL4 (13h) register to 1.
3. Set the XL_ONLY_FIFO bit in the FIFO_WTM (16h) register to 1.
4. Set the ST_SIGN_X and ST_SIGN_Y bits in the CTRL3 (12h) register to 0 and the ST_SIGN_Z bit in the WAKE_UP_DUR (1Dh) register to 1.
5. Set the FIFO_CTRL (15h) register to 00h to empty the FIFO.
6. Set the ST[1:0] bits in the SELF_TEST (32h) register to 10.
7. Set ODR = 200 Hz, BW = ODR/2, FS = ±8 *g* from the CTRL5 (14h) register and wait 100 ms.
8. Set the FIFO_CTRL (15h) register to 01h to start filling the FIFO and wait 25 ms so that at least 5 accelerometer data samples are stored in the FIFO.
9. Read the first 5 samples from FIFO whose tag refers to accelerometer-only data, compute the average for each axis, and save the result in OUT1.
10. Set the device in soft power-down and wait 10 ms.
11. Set the FIFO_CTRL (15h) register to 00h to empty the FIFO.
12. Set the ST[1:0] bits in the SELF_TEST (32h) register to 01.
13. Set ODR = 200 Hz, BW = ODR/2, FS = ±8 *g* from the CTRL5 (14h) register and wait 100 ms.
14. Set the FIFO_CTRL (15h) register to 01h to start filling the FIFO and wait 25 ms so that at least 5 accelerometer data samples are stored in the FIFO.
15. Read the first 5 samples from FIFO whose tag refers to accelerometer-only data, compute the average for each axis, and save the result in OUT2.
16. Set the ST[1:0] bits in the SELF_TEST (32h) register to 00.
17. Set the device in soft power-down and wait 10 ms.
18. Self-test deviation is 2 * |OUT2 - OUT1|. Compute the value for each axis and verify that it falls within the range provided in the datasheet.

Note: 1. Keep the device still during the self-test procedures.

2. Refer to the datasheet for the minimum and maximum values of the expected self-test difference.

Revision history

Table 46. **Document revision history**

Date	Version	Changes
03-Feb-2023	1	Initial release
02-Mar-2023	2	Updated Section 5.2: Wake-up interrupt
18-Oct-2023	3	Added footnote (3) to FUNC_CFG_ACCESS (3Fh) in Section 2: Registers Updated bit 0 of register 32h in Table 2. Main page registers Updated description and Notes in Section 2.1: Embedded functions registers and Section 2.2: Embedded advanced features registers Updated Section 3.1: Power-up sequence, Section 3.1.1: Power-up command, and Section 3.4: One-shot Updated Section 4.1: Startup sequence and Section 4.4: Using the block data update (BDU) feature Updated Section 4.5: Understanding output data and Section 4.5.1: Example of output data Updated Section 5: Interrupt generation, Section 5.2: Wake-up interrupt, Section 5.6: Activity/inactivity recognition, and Section 5.7: Boot status Updated Section 6.1: Pedometer functions: step detector and step counter and Section 6.4: Timestamp Updated description in Section 7.1: FIFO description and batched sensors Updated Section 7.2.4: EMB_FUNC_FIFO_EN Updated Section 7.6.1: MLC-generated sensors and Section 7.6.2: FSM-generated sensors Updated Section 8: Temperature sensor and Section 8.1: Example of temperature data calculation Updated Section 9: Analog hub and Qvar sensor
10-Apr-2024	4	Added external clock function to pin 7 Added register EXT_CLK_CFG (08h), changed name of register EN_DEVICE_CONFIG (3Eh), and updated the TIMESTAMP registers in Table 2. Main page registers Updated Section 4.2: Using the status register Updated Section 4.4: Using the block data update (BDU) feature Updated Section 6.4: Timestamp Minor textual updates
19-Sep-2024	5	Added SMART_POWER_CTRL (D2h) register in Section 2.2: Embedded advanced features registers and updated Section 6: Embedded functions Updated Section 3.3: Continuous conversion Added Section 3.5: External oscillator Updated Section 5.2: Wake-up interrupt Updated Section 5.3: Free-fall interrupt Updated Section 5.4.1: 6D orientation detection Updated Section 5.5.4: Tap recognition example Updated Section 5.6: Activity/inactivity recognition Updated Section 6.1: Pedometer functions: step detector and step counter Updated Section 7.1: FIFO description and batched sensors Updated Section 7.4: Main sensors Updated Section 7.5: Auxiliary sensors Updated Section 7.7: FIFO modes
02-Dec-2024	6	Updated Section 7.4: Main sensors Updated Section 10: Self-test

Contents

List of tables

List of figures