AN4712
Application note

LIS2DG: always-on 3D accelerometer

Introduction

This document is intended to provide usage information and application hints related to ST's LIS2DG motion sensor.

The LIS2DG is a 3D digital accelerometer system-in-package with a digital I^2C/SPI serial interface standard output, performing at 150 µA in high-resolution mode and no more than 80 µA in low-power mode. Thanks to the ultra-low noise performance of the accelerometer, the device combines always-on low-power features with superior sensing precision for an optimal motion experience for the consumer. Furthermore, the accelerometer features smart sleep-to-wakeup (activity) and return-to-sleep (inactivity) functions that allow advanced power saving.

The device has a dynamic user-selectable full-scale acceleration range of ±2/±4/±8/±16 *g* and is capable of measuring accelerations with output data rates from 1 Hz to 6400 Hz.

The LIS2DG can be configured to generate interrupt signals by using hardware recognition of free-fall events, 6D orientation, tap and double-tap sensing, activity or inactivity, and wake-up events.

The LIS2DG is compatible with the requirements of Android KitKat, Android L and the leading OSs, offering real and virtual sensors. It has been designed to implement in hardware significant motion, tilt, pedometer functions, and timestamp.

The LIS2DG has an integrated 256-level first-in, first-out (FIFO) buffer allowing the user to store data in order to limit intervention by the host processor.

The LIS2DG is available in a small thin plastic land grid array package (LGA) and it is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

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

Contents

List of tables

List of figures

1 Registers

Table 1: Registers

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3
MODULE_8BIT	0Ch	MODULE_7	MODULE_6	MODULE_5	MODULE_4	MODULE_3
WHO_AM_I	0Fh	0	1	0	0	0
CTRL1	20h	ODR3	ODR2	ODR1	ODR0	FS1
CTRL2	21h	BOOT	SW_RESET	0	0	FDS_SLOPE
CTRL3	22h	ST2	ST1	TAP_X_EN	TAP_Y_EN	TAP_Z_EN
CTRL4	23h	0	INT1_S_TAP	INT1_WU	INT1_FF	INT1_TAP
CTRL5	24h	DRDY _PULSED	INT2_BOOT	INT2_ON _INT1	INT2_TILT	INT2_SIG_ MOT
FIFO_CTRL	25h	FMODE2	FMODE1	FMODE0	-	MODULE _TO_FIFO
OUT_TEMP	26h	Temp7	Temp6	Temp5	Temp4	Temp3
STATUS_REG	27h	FIFO_THS	WU_IA	SLEEP _STATE	DOUBLE _TAP	SINGLE _TAP
OUTX_L	28h	D7	D6	D5	D4	D3
OUTX_H	29h	D15	D14	D13	D12	D11
OUTY_L	2Ah	D7	D6	D5	D4	D3
OUTY_H	2Bh	D15	D14	D13	D12	D11
OUTZ_L	2Ch	D7	D6	D5	D4	D3
OUTZ_H	2Dh	D15	D14	D13	D12	D11
FIFO_THS	2Eh	FTH7	FTH6	FTH5	FTH4	FTH3
FIFO_SRC	2Fh	FTH	FIFO_OVER _RUN	DIFF8	-	-

Register name	Address	Bit7	Bit6	Bit5	Bit4	Bit3
FIFO_SAMPLES	30h	DIFF7	DIFF6	DIFF5	DIFF4	DIFF3
TAP_6D_THS	31h	4D_EN	6D_THS1	6D_THS0	TAP_THS4	TAP_THS3
INT_DUR	32h	LAT3	LAT2	LAT1	LAT0	QUIET1
WAKE_UP_THS	33h	SINGLE_ DOUBLE_TAP	SLEEP_ON	WK_THS5	WK_THS4	WK_THS3
WAKE_UP_DUR	34h	FF_DUR5	WAKE_ DUR1	WAKE_ DUR0	INT1_FIFO_ FULL	SLEEP_ DUR3
FREE_FALL	35h	FF_DUR4	FF_DUR3	FF_DUR2	FF_DUR1	FF_DUR0
STATUS_DUP	36h	OVR	WU_IA	SLEEP _STATE	DOUBLE _TAP	SINGLE_ TAP
WAKE_UP_SRC	37h	-	-	FF_IA	SLEEP_ STATE_IA	WU_IA
TAP_SRC	38h	-	TAP_IA	SINGLE _TAP	DOUBLE _TAP	TAP_SIGN
6D_SRC	39h	-	6D_IA	ZH	ZL	YH
STEP_ COUNTER_MINTHS	3Ah	RST_NSTEP	PEDO4g	SC_MTHS5	SC_MTHS4	SC_MTHS3
STEP_ COUNTER_L	3Bh	STEP_COU NTER_L_7	STEP_COU NTER_L_6	STEP_COU NTER_L_5	STEP_COU NTER_L_4	STEP_COU NTER_L_3
STEP_ COUNTER_H	3Ch	STEP_COU NTER_H_7	STEP_COU NTER_H_6	STEP_COU NTER_H_5	STEP_COU NTER_H_4	STEP_COU NTER_H_3
FUNC_CK_GATE	3Dh	TILT_INT	FS_SRC1	FS_SRC0	SIG_MOT_ DET	RST_SIG_ MOT
FUNC_SRC	3Eh	-	-	-	-	-
FUNC_CTRL	3Fh	-	-	MODULE_ ON	TILT_ON	TUD_EN

2 Operating modes

The LIS2DG device provides two power modes: high-resolution (HR)/high-frequency mode (HF) and low-power (LP) mode.

After the power supply is applied, the LIS2DG performs a 20 ms boot procedure to load the trimming parameters. After the boot is completed, the accelerometer is automatically configured in power-down mode.

Referring to the LIS2DG datasheet, the output data rate (ODR) and high-frequency (HF_ODR) bits of CTRL1 register are used to select the power mode and the output data rate of the accelerometer sensor (*Table 2: "Accelerometer ODR and power mode selection"*).

Table 2: Accelerometer ODR and power mode selection

ODR [3:0]	HF_ODR	Mode	ODR selection [Hz]	Resolution (bit number)
0000	0	Power-down	Power-down	-
1000	0	LP	1	10
1001	0	LP	12.5	10
1010	0	LP	25	10
1011	0	LP	50	10
1100	0	LP	100	10
1101	0	LP	200	10
1110	0	LP	400	10
1111	0	LP	800	10
0001	0	HR	12.5	14
0010	0	HR	25	14
0011	0	HR	50	14
0100	0	HR	100	14
0101	0	HR	200	14
0110	0	HR	400	14
0111	0	HR	800	14
0101	1	HF	1600	12
0110	1	HF	3200	12
0111	1	HF	6400	12

The output data have different resolution and are left-aligned. For example in case of the 10-bit resolution the output data are the 10 most significant bits of OUT_H & OUT_L concatenation, and the raw value has to be right-shifted by 6.

Table 3: "Power consumption" shows the typical values of LIS2DG power consumption for the different operating modes.

Table 3: Power consumption

ODR [Hz]	HR/HF (µA)	LP (µA)
1	150	2.5
12.5	150	4
25	150	5.5
50	150	8
100	150	12.5
200	150	22
400	150	41
800	150	80
1600	150	-
3200	150	-
6400	150	-

2.1 Power-down

When the accelerometer is in power-down, almost all internal blocks of the device are switched off to minimize power consumption. Digital interfaces (I^2C and SPI) are still active to allow communication with the device. The content of the configuration registers is preserved and the output data registers are not updated, keeping the last data sampled in memory before going into power-down.

2.2 High-resolution/high-frequency mode

In HR/HF mode, all accelerometer circuitry is always on and data are generated at the data rate selected through the ODR bits. Data interrupt generation is active.

HR mode works with a 14-bit resolution, while HF with a 12-bit resolution (see *Table 2: "Accelerometer ODR and power mode selection"*).

2.3 Low-power mode

In low-power mode the accelerometer circuitry is periodically turned on/off with a duty cycle that is a function of the selected ODR. This mode differs from HR/HF mode in the available output data rates. In low-power mode we have same data rates as HR mode (from 12.5 Hz to 800 Hz, but with a lower consumption) plus the 1 Hz case.

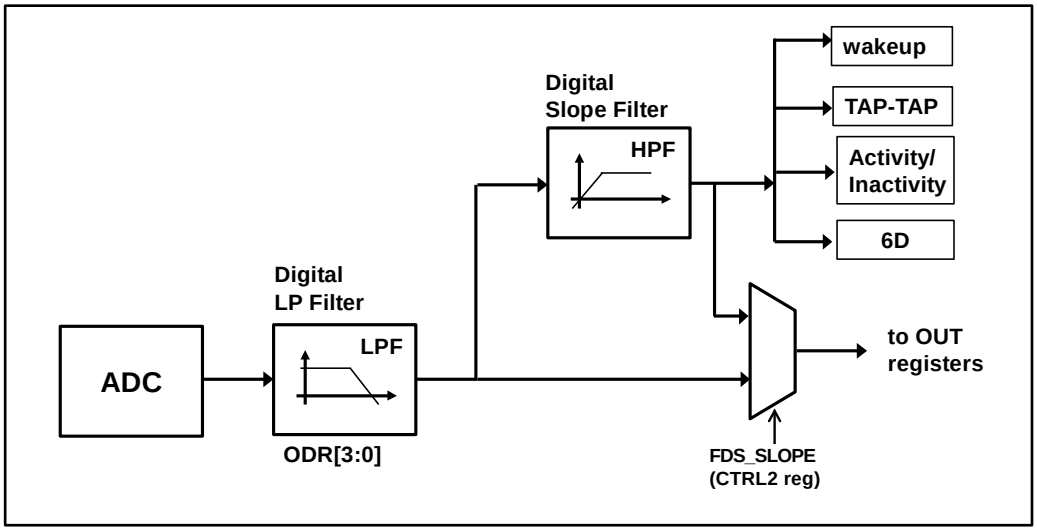
Data interrupt generation is active.

LP mode works with a 10-bit resolution (see *Table 2: "Accelerometer ODR and power mode selection"*).

2.5 Accelerometer bandwidth

The accelerometer sampling chain (*Figure 1: "Accelerometer sampling chain diagram"*) is represented by a cascade of a few blocks: an ADC converter, a digital low-pass filter and a digital slope filter.

Figure 1: Accelerometer sampling chain diagram



The digital signal is filtered by a low-pass digital filter (LPF) whose cutoff frequency depends on the selected accelerometer ODR, as shown in *Table 4: "Accelerometer LPF1 cutoff frequency"*.

Table 4: Accelerometer LPF1 cutoff frequency

Mode	ODR selection [Hz]	LPF cutoff [Hz]
LP	1	3200
LP	12.5	3200
LP	25	3200
LP	50	3200
LP	100	3200
LP	200	3200
LP	400	3200
LP	800	3200
HR	12.5	5.5
HR	25	11
HR	50	22
HR	100	44
HR	200	88
HR	400	177
HR	800	355
HF	1600	710
HF	3200	1420
HF	6400	2840

Then, the digital signal follows two paths: one towards the high-pass filter (HPF) and the other towards the ADC offset compensation. The selection of which signal (LPF or HPF) is sent to the OUT registers is determined by a multiplexer that depends on the FDS_SLOPE bit of CTRL2 register. When it is logic '1', the HPF signal is selected, LPF otherwise.

The signal that is sent to the digital functions (wakeup, tap-tap, activity/inactivity and 6D orientation) is always the HPF signal.

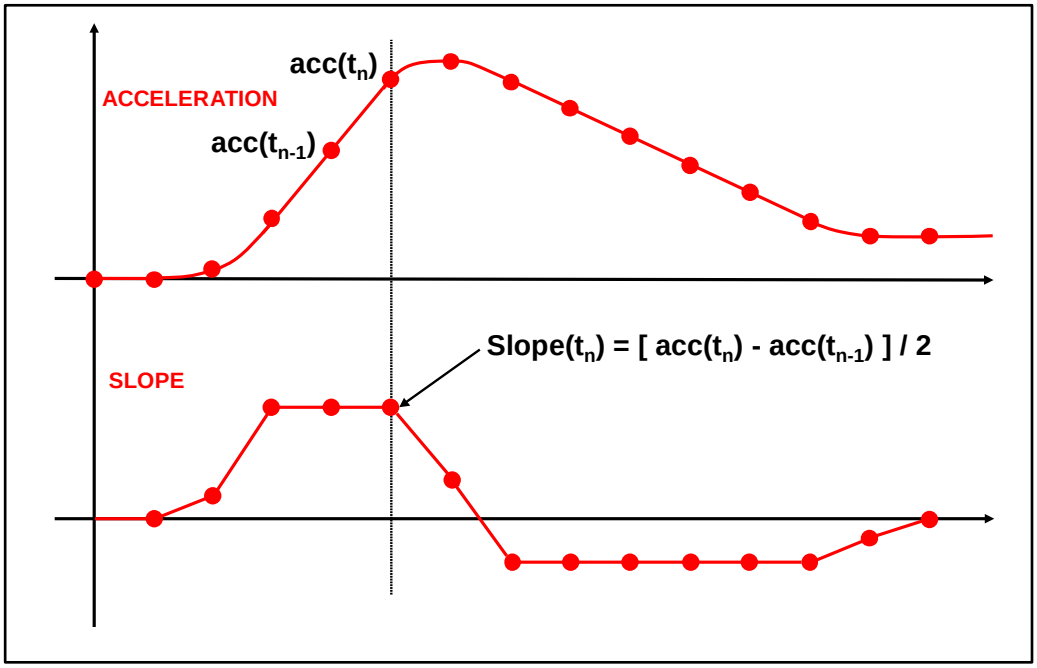
2.5.1 Accelerometer slope filter

As shown in *Figure 2: "Accelerometer slope filter"*, the LIS2DG device embeds a digital slope filter which is used for wakeup and single/double-tap features. 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 *Figure 2: "Accelerometer slope filter"*.

Figure 2: Accelerometer slope filter



3 Reading output data

3.1 Startup sequence

Once the device is powered up, it automatically downloads the calibration coefficients from the embedded flash to the internal registers. When the boot procedure is completed, i.e. after approximately 20 milliseconds, the accelerometer automatically enters power-down.

To turn on the accelerometer and gather acceleration data, it is necessary to select one of the operating modes through the CTRL1 register.

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

1. Write CTRL1 = 60h // Acc = 400 Hz (high-resolution mode)
2. Write CTRL4 = 01h // Acc data-ready interrupt on INT1

3.2 Using the status register

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

For the accelerometer, the reads should be performed as follows:

1. Read STATUS
2. If DRDY = 0, then go to 1
3. Read OUTX_L
4. Read OUTX_H
5. Read OUTY_L
6. Read OUTY_H
7. Read OUTZ_L
8. Read OUTZ_H
9. Data processing
10. Go to 1

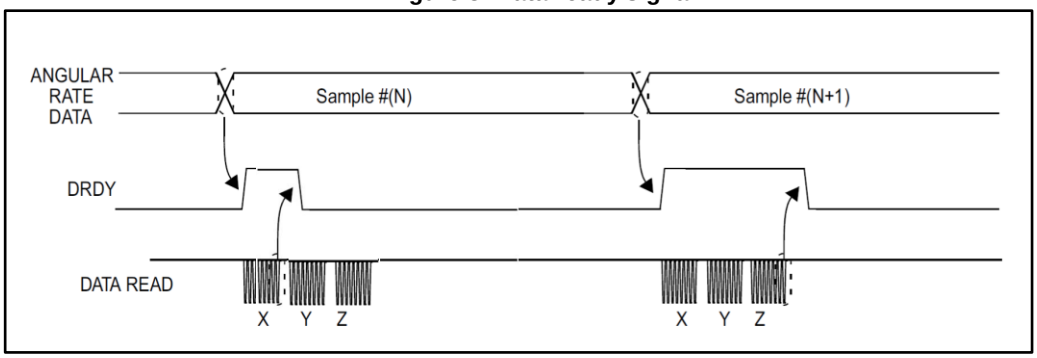
3.3 Using the data-ready signal

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

For the accelerometer sensor, the data-ready signal is represented by the DRDY bit of the STATUS_REG register. The signal can be driven to the INT1 pin by setting to 1 the INT1_DRDY bit of the CTRL4 register and to the INT2 pin by setting to 1 the INT2_DRDY bit of the CTRL5 register.

The data-ready signal rises to 1 when a new set of data has been generated and it is available for reading. In DRDY latched mode (DRDY_PULSED bit set to 0 in CTRL5 register), which is the default condition, the signal gets reset when the higher part of one of the channels has been read (29h, 2Bh, 2Dh).

Figure 3: Data-ready signal



3.4 Using the block data update (BDU) feature

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

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

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

3.5 Understanding output data

The measured acceleration data are sent to the OUTX_H, OUTX_L, OUTY_H, OUTY_L, OUTZ_H, and OUTZ_L registers. These registers contain, respectively, the most significant part and the least significant part of the acceleration signals acting on the X, Y, and Z axes.

The complete output data for the X, Y, Z channels is given by the concatenation OUTX_H & OUTX_L, OUTY_H & OUTY_L , OUTZ_H & OUTZ_L and it is expressed as a two's complement number.

Acceleration data is represented as 16-bit numbers, called LSB, but has different resolution according to the selected operating mode (LP/HR/HF). See *Table 2: "Accelerometer ODR and power mode selection"*.

Since the acceleration data is always aligned to the most significant bits, the 16-bit LSB number has to be right-shifted according to the selected resolution:

- LP mode (10-bit resolution): right-shift-by-6
- HF mode (12-bit resolution): right-shift-by-4
- HR mode (14-bit resolution): right-shift-by-2

After calculating the LSB, it must be multiplied by the proper sensitivity parameter to obtain the corresponding value in mg.

3.5.2 Example of output data

Hereafter there 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 perfect device calibration (i.e. no offset, no gain error,....).

Get raw data from the sensor (HR mode, ODR 200 Hz):

```
OUTX_L: 5Ch
OUTX_H: FDh
OUTY_L: 74h
OUTY_H: 00h
OUTZ_L: F8h
OUTZ_H: 42h
```

Do registers concatenation:

```
OUTX_H & OUTX_L: FD5Ch
OUTY_H & OUTY_L: 0074h
OUTZ_H & OUTZ_L: 42F8h
```

Calculate signed decimal value (two’s complement format) and right-shift it by 2.

```
X: -169
Y: +29
Z: +4286
```

Apply sensitivity (e.g. 0.244 at full scale 2 g):

```
X: -169 * 0.244 = -41mg
Y: +29 * 0.244 = +7mg
Z: +4286 * 0.244 = +1046mg
```

4 Interrupt generation and Android embedded functions

In the LIS2DG device the interrupt generation is based on accelerometer data, so, for interrupt-generation purposes, the accelerometer sensor has to be set in an active operating mode (not in power-down).

The interrupt generator can be configured to detect also:

- Free-fall;
- Wake-up;
- 6D/4D orientation detection;
- Single-tap and double-tap sensing;
- Activity/Inactivity detection.

In addition, the LIS2DG can efficiently run the sensor-related features specified in Android, saving power and enabling faster reaction time. In particular, it has been designed to implement in hardware:

- Significant motion;
- Tilt;
- Pedometer functions;

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

The H_LACTIVE bit of the CTRL3 register must be used to select the polarity of the interrupt pins. If this bit is set to 0 (default value), the interrupt pins are active high and they change from low to high level when the related interrupt condition is verified. Otherwise, if the H_LACTIVE bit is set to 1 (active low), the interrupt pins are normally at high level and they change from high to low when interrupt condition is reached.

The PP_OD bit of CTRL3 allows changing the behavior of the interrupt pins from push-pull to open drain. If the PP_OD bit is set to 0, the interrupt pins are in push-pull configuration (low-impedance output for both high and low level). When the PP_OD bit is set to 1, only the interrupt active state is a low-impedance output.

The LIR bit of CTRL3 allows applying the latched mode to the interrupt signals. When the LIR bit is set to 1, once the interrupt pin is asserted, it must be reset by reading the related interrupt source register. If the LIR bit is set to 0, the interrupt signal is automatically reset when the interrupt condition is no longer verified or after a certain amount of time.

4.1 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 CTRL4 register for the INT1 pin, and through the CTRL5 register for the INT2 pin.

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

Table 5: CTRL4 register

b7	b6	b5	b4	b3	b2	b1	b0
0	INT1_ S_TAP	INT1_ WU	INT1_ FF	INT1_ TAP	INT1_ 6D	INT1_ FTH	INT1_ DRDY

- INT1_S_TAP: Single-tap event recognition is routed on the INT1 pad.
- INT1_WU: Wakeup event recognition is routed on the INT1 pad.
- INT1_FF: Free-fall event recognition is routed on the INT1 pad.
- INT1_TAP: Tap event recognition is routed on the INT1 pad.
- INT1_6D: 6D event recognition is routed on the INT1 pad.
- INT1_FTH: FIFO threshold event is routed on the INT1 pad.
- INT1_DRDY: Accelerometer data-ready is routed on the INT1 pad.

Table 6: CTRL5 register

b7	b6	b5	b4	b3	b2	b1	b0
DRDY_ PULSED	INT2_ BOOT	INT2_ON _INT1	INT2_ TILT	INT2_SIG _MOT	INT2_STEP _DET	INT2_ FTH	INT2_ DRDY

- DRDY_PULSED: Data-ready interrupt mode selection: latched mode / pulsed mode.
- INT2_BOOT: Boot state routed on the INT2 pad.
- INT2_ON_INT1: All INT2 signals are routed also to the INT1 pad.
- INT2_TILT: Tilt event recognition is routed on the INT2 pad.
- INT2_SIG_MOT: Significant motion event recognition is routed on the INT2 pad.
- INT2_STEP_DET: Step event recognition is routed on the INT2 pad.
- INT2_FTH: FIFO threshold event is routed on the INT2 pad.
- INT2_DRDY: Accelerometer data-ready on the INT2 pad.

4.2 Event status

If multiple interrupt signals are routed on the same pin (INTx), the logic level of this pin is the “OR” combination of the selected interrupt signals. In order to know which event has generated the interrupt condition, the application should read the proper status register, which also will clear the event.

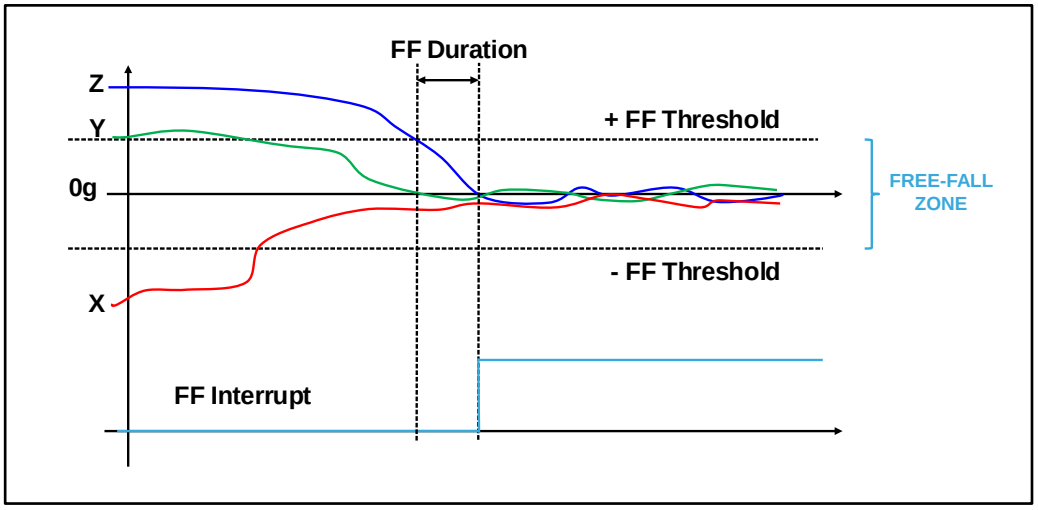
As indicated below, the STATUS register is duplicated at address 36h in order to allow a multiple read of consecutive registers (36/37h/38h/39h).

- STATUS (27h) or STATUS_DUP (36h)
- WAKE_UP_SRC (37h)
- TAP_SRC (38h)
- 6D_SRC (39h)
- FUNC_CK_GATE (3Dh).

4.3 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 case 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 4: "Free-fall interrupt"*).

Figure 4: Free-fall interrupt



The free-fall event signal can be routed to the INT1 pin by setting to 1 the INT1_FF bit of the CTRL4 register; it can also be checked by reading the FF_IA bit of the WAKE_UP_SRC register.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal is automatically reset when the free-fall condition is no longer verified. If latch mode is enabled and the free-fall interrupt signal is driven to the interrupt pins, once a free-fall event has occurred and the interrupt pin is asserted, it must be reset by reading the WAKE_UP_SRC register. If the latch mode is enabled, but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

The register used to configure the threshold parameter is named FREE_FALL; the unsigned threshold value is related to the value of the FF_THS[2:0] field value as indicated in *Table 7: "Free-fall threshold LSB value"* and is expressed in units of 31.25 mg. The LSB values given in this table are valid for any accelerometer full-scale value.

Table 7: Free-fall threshold LSB value

FREE_FALL - FF_THS[2:0]	Threshold LSB value
000	5
001	7
010	8
011	10
100	11
101	13
110	15
111	16

Duration time is measured in N/ODR, where N is the content of the FF_DUR[5:0] field of the FREE_FALL / WAKE_UP_DUR registers and ODR is the accelerometer data rate.

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

1.	Write 60h in CTRL1	// Turn on the accelerometer // ODR = 400 Hz, FS = 2 *g*
2.	Write 00h in WAKE_UP_DUR	// Set event duration (FF_DUR5 bit)
3.	Write 33h in FREE_FALL	// Set FF threshold (FF_THS[2:0] = 011b) // Set six sample event duration (FF_DUR[5:0] = 000110b)
4.	Write 10h in CTRL4	// FF interrupt driven to INT1 pin
5.	Write 04h in CTRL3	// Latch interrupt

The sample code exploits a threshold set to ~310 m*g* (31.25 m*g* * 10) for free-fall recognition and the event is notified by hardware through the INT1 pin. The FF_DUR[5:0] field of FREE_FALL / WAKE_UP_DUR registers is configured like this to ignore events that are shorter than 6/ODR = 6/400 Hz = 15 msec in order to avoid false detections.

4.4 Wake-up interrupt

In the LIS2DG device the wake-up feature is implemented using the slope filter (see *Section 2.4.1: "Accelerometer slope filter"* for more details), as illustrated in *Figure 2: "Accelerometer slope filter"*. The wake-up interrupt signal is generated if a certain number of consecutive filtered data exceed the configured threshold (*Figure 5: "Wake-up interrupt"*).

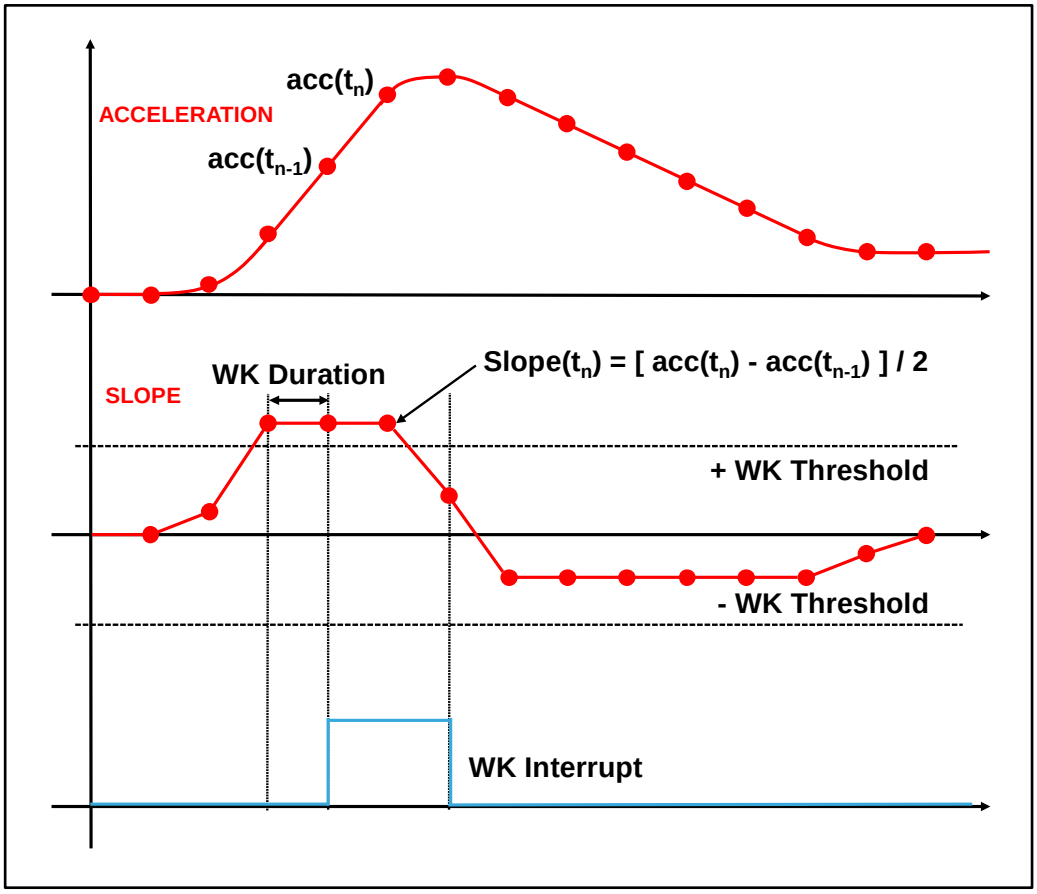
The unsigned threshold value is defined using the WK_THS[5:0] bits of the WAKE_UP_THS register; the value of 1 LSB of these 6 bits depends on the selected accelerometer full scale: 1 LSB = (FS)/(2^6). The threshold is applied to both positive and negative data: for a wake-up interrupt generation all three axes must be bigger than the threshold.

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

This interrupt signal can be driven to the INT1 interrupt pin by setting to 1 the INT1_WU bit of the CTRL4 register; it can also be checked by reading the WU_IA bit of the WAKE_UP_SRC register. The X_WU, Y_WU, Z_WU bits of the WAKE_UP_SRC register indicate which axis has triggered the wake-up event.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal is automatically reset when the filtered data falls below the threshold. If latch mode is enabled and the wake-up interrupt signal is driven to the interrupt pins, once a wake-up event has occurred and the interrupt pin is asserted, it must be reset by reading the WAKE_UP_SRC register. If the latch mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

Figure 5: Wake-up interrupt



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

1.	Write 60h in CTRL1	// Turn on the accelerometer // ODR = 400 Hz, FS = 2 *g*
2.	Write 00h in WAKE_UP_DUR	// No duration
3.	Write 02h in WAKE_UP_THS	// Set wake-up threshold
4.	Write 20h in CTRL4	// Wake-up interrupt driven to INT1 pin

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

4.5 6D/4D orientation detection

The LIS2DG 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.

4.5.1 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 re-asserted as long as the position is maintained.

6D interrupt is generated when, for two consecutive samples, 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 6D_SRC (39h) register indicate which axis has triggered the 6D event.

In more detail:

Table 8: 6D_SRC register

b7	b6	b5	b4	b3	b2	b1	b0
0	6D_IA	ZH	ZL	YH	YL	XH	XL

- 6D_IA is set high when the device switches from one orientation to another.
- 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 module 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 module bigger than the threshold.

The 6D_THS[1:0] bits of the TAP_6D_THS register are used to select the threshold value used to detect the change in device orientation. The threshold values given in *Table 9: "Threshold for 4D/6D function"* are valid for each accelerometer full-scale value.

Table 9: Threshold for 4D/6D function

6D_THS[1:0]	Threshold value [degrees]
00	80
01	70
10	60
11	50

This interrupt signal can be driven to the INT1 interrupt pin by setting to 1 the INT1_6D bit of the CTRL4 register; it can also be checked by reading the 6D_IA bit of the 6D_SRC register.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal is active only for 1/ODR[s] then it is automatically deasserted (ODR is the accelerometer output data rate). If latch mode is enabled and the 6D interrupt signal is driven to the interrupt pins, once an orientation change has occurred and the interrupt pin is asserted, a reading of the STATUS_DUP register clears the request and the device is ready to recognize a different orientation. If the latch mode is enabled, but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

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

Figure 6: 6D recognized orientations

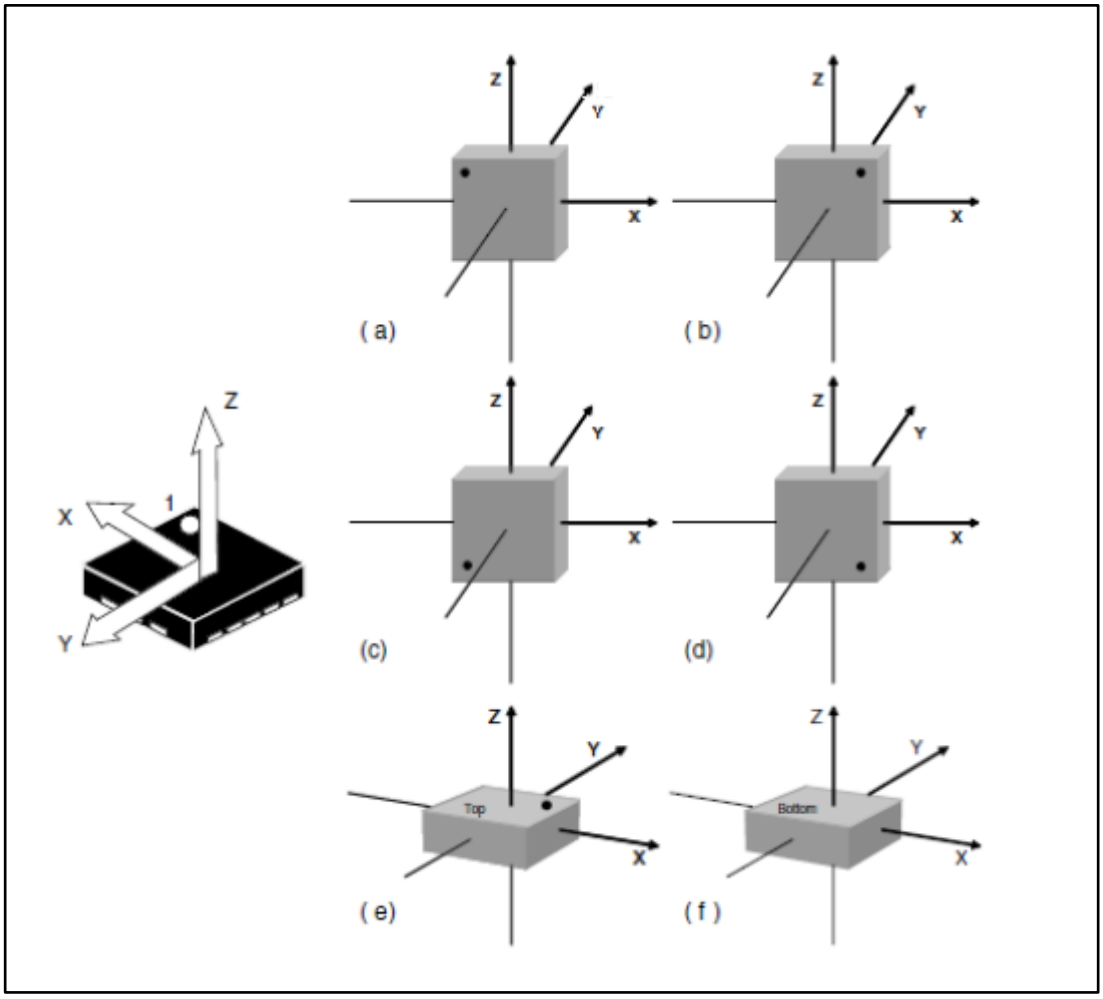


Table 10: 6D_SRC register for 6D positions

Case	6D_IA	ZH	ZL	YH	YL	XH	XL
(a)	1	0	0	0	1	0	0
(b)	1	0	0	0	0	1	0
(c)	1	0	0	0	0	0	1
(d)	1	0	0	1	0	0	0
(e)	1	1	0	0	0	0	0
(f)	1	0	1	0	0	0	0

Hereafter an example which implements the SW routine for 6D orientation detection:

1.	Write 60h in CTRL1	// Turn on the accelerometer // ODR = 400 Hz, FS = 2 *g*
2.	Write 40h in TAP_6D_THS	// Set 6D threshold (6D_THS[1:0] = 10b = 60 degrees)
3.	Write 04h in CTRL4	// 6D interrupt driven to INT1 pin

4.5.2 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 4D_EN bit of the TAP_6D_THS register to 1. In this configuration, the Z-axis position detection is disabled, therefore reducing position recognition to cases (a), (b), (c), and (d) of *Table 10: "6D_SRC register for 6D positions"*.

4.6 Single-tap and double-tap recognition

The single-tap and double-tap recognition functions featured in the LIS2DG help to create a man-machine interface with little software loading. The device can be configured to output an interrupt signal on a dedicated pin when tapped in any direction.

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

In the LIS2DG device the single-tap and double-tap recognition functions use the slope between two consecutive acceleration samples to detect the tap events; the slope data is computed using the following formula:

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

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

Single and double-tap recognition is meaningful only for ODR ≥ 400 Hz.

4.6.1 Single tap

If the device is configured for single-tap event detection, an interrupt is generated when the slope data on the selected channel exceed the programmed threshold, and return below it within the shock time window.

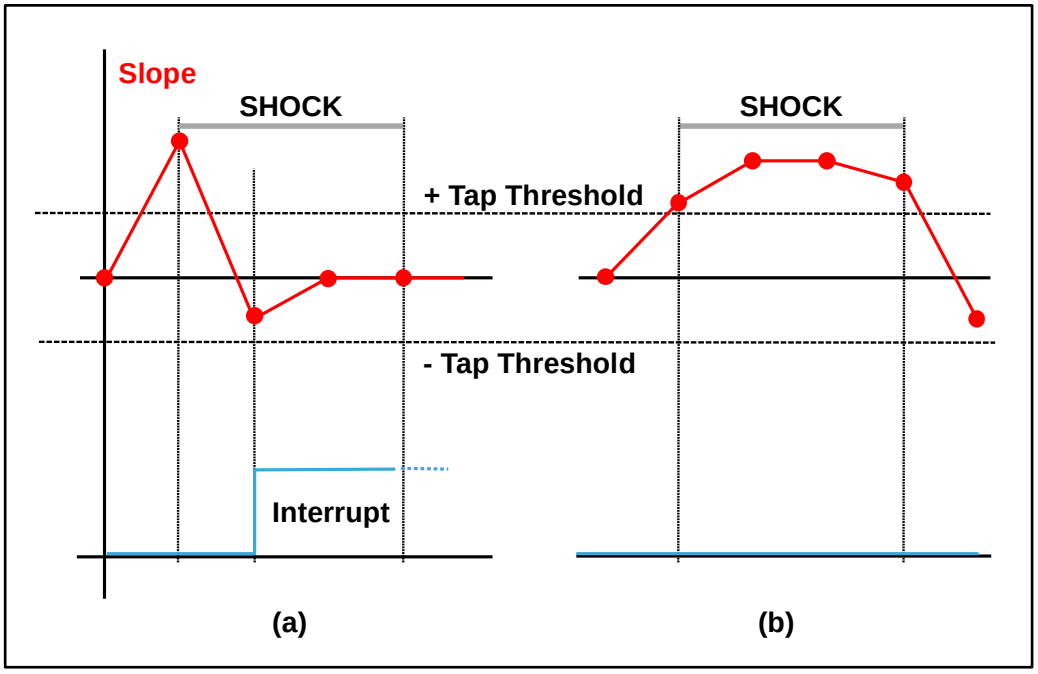
In the single-tap case, if the LIR bit of the CTRL3 register is set to 0, the interrupt is kept high for the duration of the quiet window.

In order to enable the latch feature on the single-tap interrupt signal, the LIR bit of CTRL3 has to be set to 1: the interrupt is kept high until the TAP_SRC register is read.

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

In case (a) of *Figure 7: "Single-tap event recognition"* the single-tap event has been recognized, while in case (b) the tap has not been recognized because the slope data fall under the threshold after the shock time window has expired.

Figure 7: Single-tap event recognition



4.6.2 Double tap

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

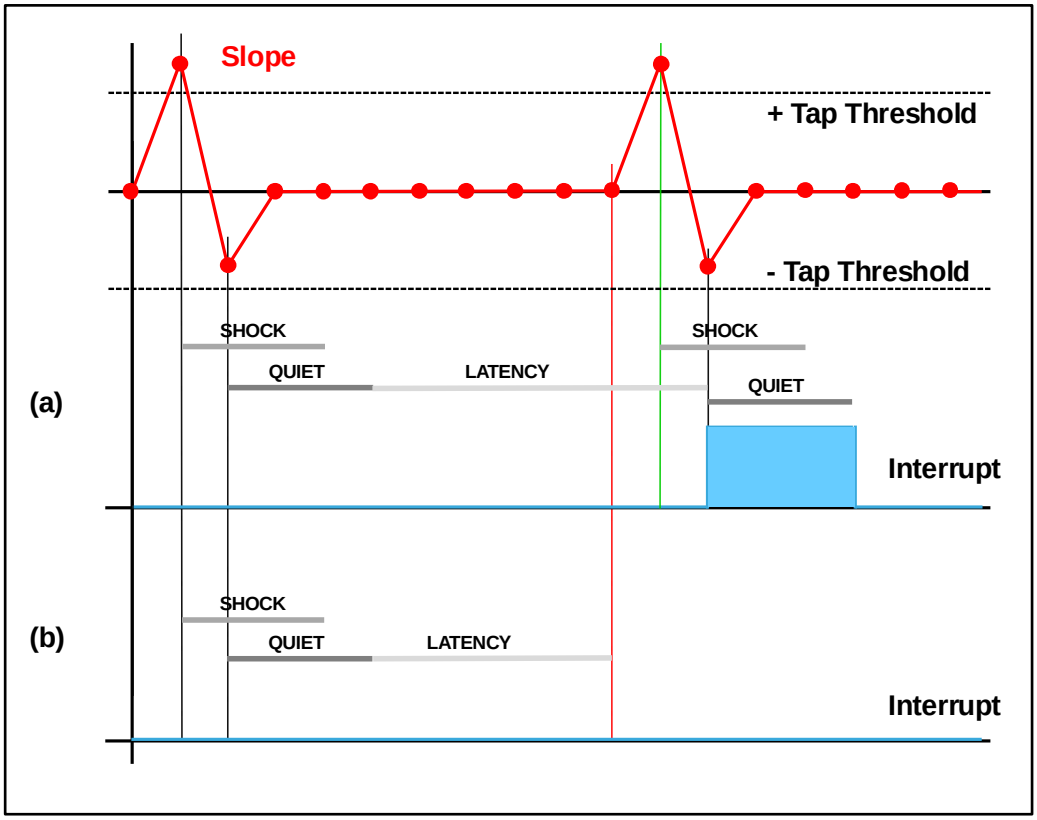
In particular, after the first tap has been recognized, the second tap detection procedure is delayed for an interval defined by the quiet time. This means that after the first tap has been recognized, the second tap detection procedure starts only if the slope data exceed the threshold after the quiet window but before the latency window has expired. In case (a) of *Figure 8: "Double-tap event recognition (LIR bit = 0)"*, a double-tap event has been correctly recognized, while in case (b) the interrupt has not been generated because the slope data exceed the threshold after the latency window interval has expired.

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

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

In the double-tap case, if the LIR bit of the TAP_CFG register is set to 0, the interrupt is kept high for the duration of the quiet window. If the LIR bit is set to 1, the interrupt is kept high until the TAP_SRC register is read.

Figure 8: Double-tap event recognition (LIR bit = 0)



4.6.3 Single-tap and double-tap recognition configuration

The LIS2DG device can be configured to output an interrupt signal when tapped (once or twice) in any direction: the TAP_X_EN, TAP_Y_EN and TAP_Z_EN bits of the TAP_CFG register must be set to 1 to enable the tap recognition on X, Y, Z directions, respectively.

Configurable parameters for tap recognition functionality are the tap threshold and the shock, quiet and latency time windows. Valid ODRs are 400 Hz, 800 Hz and 1600 Hz.

The TAP_THS[4:0] bits of the TAP_6D_THS register are used to select the unsigned threshold value used to detect the tap event. The value of 1 LSB of these 5 bits depends on the selected accelerometer full scale: 1 LSB = (FS)/(2^5). The unsigned threshold is applied to both positive and negative slope data.

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

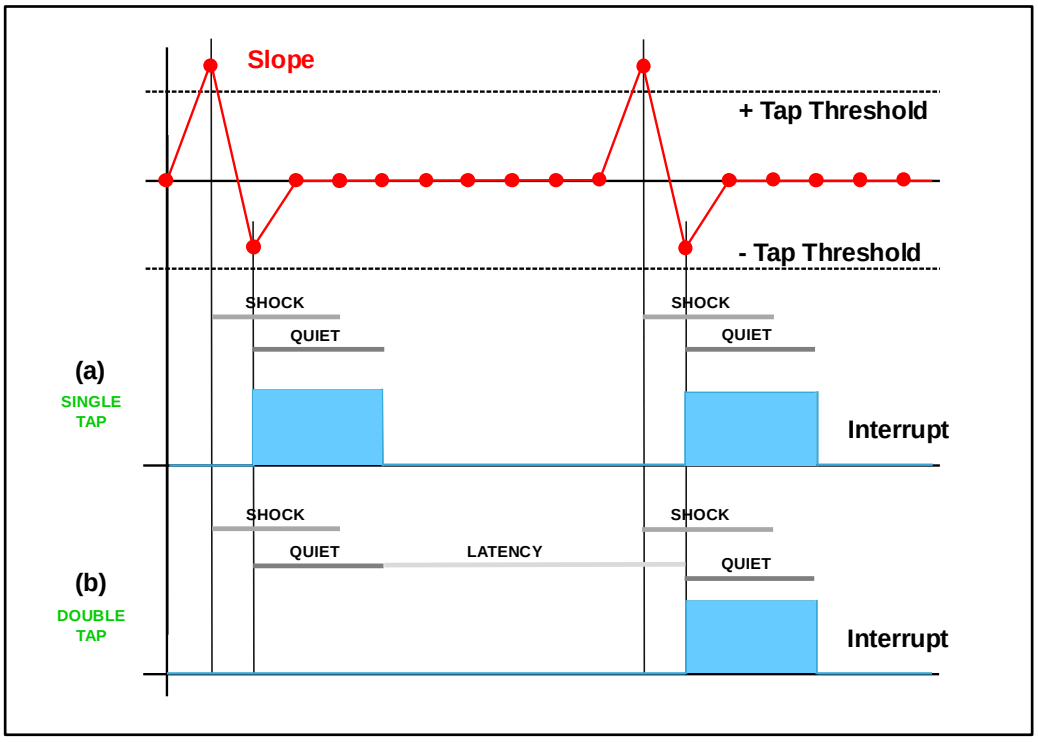
In the double-tap case, the quiet time window defines the time after the first tap recognition in which there must not be any overthreshold. When the latch mode is disabled (LIR bit of TAP_CFG is set to 0), the quiet time also defines the length of the interrupt pulse (in both single and double-tap case).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

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

Figure 9: "Single and double-tap recognition (LIR bit = 0)" illustrates a single-tap event (a) and a double-tap event (b). These interrupt signals can be driven to the INT1 interrupt pin by setting to 1 the INT1_S_TAP bit of the CTRL4 register for the single-tap case, and setting to 1 the INT1_TAP bit of the CTRL4 register for the double-tap case.

No single/double-tap interrupt is generated if the accelerometer is in inactivity status (see *Section 4.7: "Activity/Inactivity recognition"* for more details).

Figure 9: Single and double-tap recognition (LIR bit = 0)



The tap interrupt signals can also be checked by reading the TAP_SRC (38h) register, described in *Table 11: "TAP_SRC register"*.

Table 11: TAP_SRC register

b7	b6	b5	b4	b3	b2	b1	b0
0	TAP_IA	SINGLE _TAP	DOUBLE _TAP	TAP _SIGN	X_TAP	Y_TAP	Z_TAP

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

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

If the latch mode is enabled and the interrupt signal is driven to the interrupt pins, the value assigned to SINGLE_DOUBLE_TAP also affects the behavior of the interrupt signal: when it is set to 0, the latch mode is applied to the single-tap interrupt signal; when it is set to 1, the latch mode is applied to the double-tap interrupt signal only. The latched interrupt signal is kept high until the TAP_SRC register is read. If the latch mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

4.6.4 Single-tap example

Hereafter an example code which implements the SW routine for single-tap detection.

1.	Write 60h in CTRL1	// Turn on the accelerometer // ODR = 400 Hz, FS = 2 *g*
2.	Write 38h in CTRL3	// Enable tap detection on X, Y, Z axis
3.	Write 09h in TAP_6D_THS	// Set tap threshold
4.	Write 06h in INT_DUR	// Set quiet and shock time windows
5.	Write 00h in WAKE_UP_THS	// Only single tap enabled (SINGLE_DOUBLE_TAP = 0)
6.	Write 40h in CTRL4	// Single tap interrupt driven to INT1 pin

In this example the TAP_THS field of the TAP_6D_THS register is set to 01001b, therefore the tap threshold is 562.5 mg (= 9 * FS / 2^5).

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

The QUIET field of the INT_DUR register is set to 01b: since the latch mode is disabled, the interrupt is kept high for the duration of the quiet window, therefore 9.6 ms (= 1 * 8 / ODR).

4.6.5 Double-tap example

The example code which implements the SW routine for single-tap detection is given below.

1.	Write 60h in CTRL1	// Turn on the accelerometer // ODR = 400 Hz, FS = 2 *g*
2.	Write 38h in TAP_CFG	// Enable tap detection on X, Y, Z axis
3.	Write 0Ch in TAP_6D_THS	// Set tap threshold
4.	Write 7Fh into INT_DUR	// Set duration, quiet and shock time windows
5.	Write 80h in WAKE_UP_THS	// Single & double-tap enabled (SINGLE_DOUBLE_TAP = 1)
6.	Write 08h in CTRL4	// Double-tap interrupt driven to INT1 pin

In this example the TAP_THS field of the TAP_6D_THS register is set to 01100b, therefore the tap threshold is 750 mg (= 12 * FS / 2^5).

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

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

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

4.7 Activity/Inactivity recognition

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

When the activity/inactivity recognition function is activated, the LIS2DG device is able to automatically enter low-power mode and decrease the accelerometer sampling rate to

12.5 Hz, increasing back the accelerometer ODR and bandwidth as soon as the wake-up interrupt event has been detected.

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.

The activity/inactivity recognition function is enabled by setting to 1 the SLEEP_ON bit of the WAKE_UP_THS register.

The activity/inactivity recognition function uses the slope between two consecutive acceleration samples to detect the activity/inactivity event; the slope data is computed using the following formula:

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

This function can be fully programmed by the user in terms of expected amplitude and timing of the slope data by means of a dedicated set of registers (*Figure 10: "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: 1 LSB = (FS)/(2^6). The threshold is applied to both positive and negative slope data.

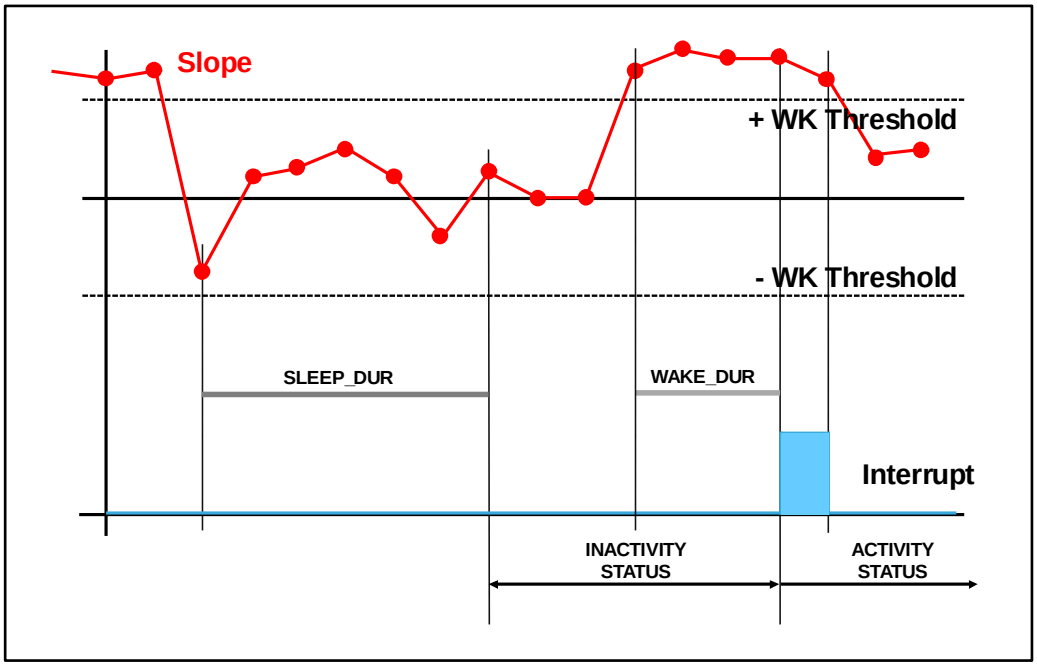
When a certain number of consecutive X,Y,Z slope data is smaller than the configured threshold, the ODR [3:0] bits of the CTRL1 register are bypassed (inactivity) and the accelerometer is internally set to 12.5 Hz although the content of CTRL1 is left untouched. The duration of the inactivity status to be recognized is defined by the SLEEP_DUR[3:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to 512*ODR time, where ODR is the accelerometer output data rate.

When the inactivity status is detected, no interrupt is generated to the application processor (SLEEP_STATE_IA bit of the WAKE_UP_SRC register cannot be routed on the pad).

When a certain number of consecutive slope data on one axis becomes bigger than the threshold, the CTRL1 register settings are immediately restored (activity). The duration of the activity status to be recognized is defined by the WAKE_DUR[1:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to 1*ODR time, where ODR is the accelerometer output data rate.

When the activity status is detected, the interrupt is set high for 1/ODR[s] period, then it is automatically deasserted (the WU_IA event on the pad must be routed by setting the INT1_WU bit of CTRL4 register to 1).

Figure 10: Activity/Inactivity recognition



The code provided below is a basic routine for activity/inactivity detection implementation.

1.	Write 50h in CTRL1	// Turn on the accelerometer // ODR = 200 Hz, FS = 2 *g*
2.	Write 42h in WAKE_UP_DUR	// Set duration for inactivity detection // Set duration for activity detection
3.	Write 42h in WAKE_UP_THS	// Set activity/inactivity threshold // Enable activity/inactivity detection
4.	Write 20h in CTRL4	// Activity (wakeup) interrupt driven to INT1 pin

In this example the WK_THS field of the WAKE_UP_THS register is set to 000010b, therefore the activity/inactivity threshold is 62.5 m*g* (= 2 * FS / 2^6).

Before inactivity detection, the X,Y,Z slope data must be smaller than the configured threshold for a period of time defined by the SLEEP_DUR field of the WAKE_UP_DUR register: this field is set to 0010b, corresponding to 4.92 s (= 2 * 512 / ODR). After this period of time has elapsed, the accelerometer ODR is internally set to 12.5 Hz.

The activity status is detected and the CTRL1 register settings immediately restored if the slope data of (at least) one axis are bigger than the threshold for a period of time defined by the WAKE_DUR field of the WAKE_UP_DUR register: this field is set to 10b, corresponding to 9.62 ms (= 2 * 1 / ODR).

4.8 Boot status

After the device is powered up, the LIS2DG performs a 20 ms boot procedure to load the trimming parameters. After the boot is completed the accelerometer is automatically configured in power-down mode.

After power-up, the trimming parameters can be re-loaded by setting to 1 the BOOT bit of the CTRL2 register.

No toggle of the device power lines is required and the content of the device control registers is not modified, so the device operating mode doesn’t change after boot. If the reset to the default value of the control registers is required, it can be performed by setting to 1 the SW_RESET bit of the CTRL2 register.

The boot status signal can be driven to the INT2 interrupt pin by setting to 1 the INT2_BOOT bit of the CTRL5 register: the signal goes to ‘1’ while a boot is taking place, and returns to ‘0’ when it is done.

To return the device to the power-down default settings, follow these steps from ANY operating mode:

1. Set SW_RESET bit to '1'
2. Wait until SW_RESET bit returns to '0'
3. Set REBOOT bit to '1'
4. Wait 5 ms

4.10 Android embedded functions

The LIS2DG device implements in hardware the sensor-related functions specified in Android L; specific IP blocks with negligible power consumption and high-level performance implement the following functions using only the accelerometer:

- Pedometer functions (step detector and step counter)
- Significant motion
- Tilt

4.10.1 Pedometer functions: step detector and step counter

A specific IP block of the LIS2DG device is dedicated to pedometer functions: the step detector and the step counter.

Pedometer functions work at 25 Hz and they are completely independent from the ODR.

In order to enable the pedometer functions the STEP_CNT_ON bit of the FUNC_CTRL register must be set to 1.

The step detector functionality generates an interrupt every time a step is recognized. In case of interspersed step sessions, 7 consecutive steps have to be detected before the first interrupt generation (debounce functionality) in order to avoid false step detections.

This interrupt signal can be driven to the INT2 interrupt pin by setting to 1 the INT2_STEP_DET bit of the CTRL5 register; it can also be checked by reading the STEP_DETECT bit of the FUNC_CK_GATE 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 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; it can be reset to zero by setting the RST_NSTEP bit of the STEP_COUNTER_MINTHS register to 1.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal generated by the pedometer functions is pulsed: the duration of the pulse observed on the interrupt pins is about 60 µs.

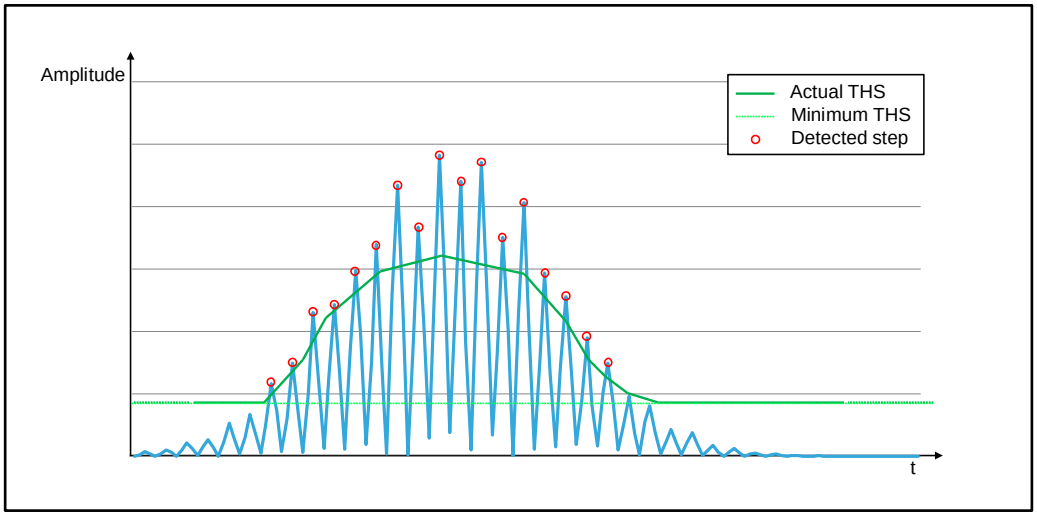
If latch mode is enabled (LIR bit of CTRL3 is set to 1) and the interrupt signal is driven to the interrupt pins, once a step has occurred, a reading of the FUNC_CK_GATE register clears the request on both the pins and the device is ready to recognize the next step. If latch mode is enabled but the event is not driven to the interrupt pins, the STEP_DETECT bit of the FUNC_CK_GATE register is pulsed, with a fixed duration of 1/25 Hz.

As default, the step counter works at 2 *g* full scale, independently of the configured device full scale, but it can be configured to 4 *g* by setting to 1 the PEDO4g bit of the STEP_COUNTER_MINTHS register.

It is also possible to set the “minimum threshold”, i.e. the value at which the threshold for step recognition asymptotically tends if no steps are detected and below which it cannot descend. This configuration is in the SC_MTHS[5:0] field of the STEP_COUNTER_MINTHS register. The value of 1 LSB of these 6 bits depends on the selected step counter full scale

(2 *g* or 4 *g*): 1 LSB = (FS)/(2^6).

Figure 11: Minimum threshold



Hereafter a basic SW routine which shows how to enable the pedometer functions:

1.	Write 20h in CTRL1	// Turn on the accelerometer // ODR = 25 Hz, FS = 2 *g*
2.	Write 01h in FUNC_CTRL	// Enable pedometer algorithm
3.	Write 04h in CTRL5	// Step detector interrupt driven to INT2 pin

The interrupt signal is generated when a step is recognized and the step count is available by reading the STEP_COUNTER_H / STEP_COUNTER_L registers.

4.10.2 Significant motion

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 generates an interrupt when a ‘significant motion’, that could be due to a change in user location, is detected. To be considered significant, a motion should be at least 5 steps.

In the LIS2DG device this function has been implemented in hardware using only the accelerometer and works at 25 Hz, independently of the device ODR.

In order to enable significant motion detection, the SIG_MOTION_ON bit of the FUNC_CTRL register must be set to 1.

The significant motion interrupt signal can be driven to the INT2 interrupt pin by setting to 1 the INT2_SIG_MOT bit of the CTRL5 register; it can also be checked by reading the SIG_MOT_DET bit of the FUNC_CK_GATE register.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal generated by the significant motion function is pulsed: the duration of the pulse observed on the interrupt pins is about 60 µs; the duration of the pulse observed on the SIG_MOT_DET bit of the FUNC_CK_GATE register is 1/25 Hz.

If latch mode is enabled (LIR bit of CTRL3 is set to 1) and the interrupt signal is driven to the interrupt pin, once a ‘significant motion’ is detected, a reading of the FUNC_CK_GATE register clears the request on INT2 and the SIG_MOT_DET bit of the FUNC_CK_GATE register, and the device is ready to recognize the next event. If latch mode is enabled but

the interrupt signal is not driven to the interrupt pins, the SIG_MOT_DET bit of the FUNC_CK_GATE register is pulsed, with a fixed duration of 1/25 Hz.

Hereafter a basic SW routine which shows how to enable the significant motion detection function:

1. Write 20h in CTRL1 // Turn on the accelerometer // ODR = 25 Hz, FS = 2 *g*
2. Write 02h in FUNC_CTRL // Enable significant motion algorithm
3. Write 08h in CTRL5 // Significant motion interrupt driven to INT2 pin

4.10.3 Tilt

The tilt function allows detecting when an activity change occurs (e.g. when phone is in a front pocket and the user goes from sitting to standing or standing to sitting): in the LIS2DG device it has been implemented in hardware using only the accelerometer.

In order to enable tilt detection the TILT_ON bit of the FUNC_CTRL register must be set to 1.

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 tilt detection is enabled or the position of the device when the last tilt interrupt was generated.

After this function is enabled, for the generation of the first tilt interrupt the device should be continuously tilted by an angle greater than 35 degrees from the start position for a period of time of at least 2 seconds. After the first tilt interrupt is generated, the tilt interrupt signal is set high as soon as the device is tilted by an angle greater than 35 degrees from the position of the device corresponding to the last interrupt detection (no need to wait 2 seconds).

This interrupt signal can be driven to the INT2 pin by setting to 1 the INT2_TILT bit of the CTRL5 register; it can also be checked by reading the TILT_INT bit of the FUNC_CK_GATE register.

If latch mode is disabled (LIR bit of CTRL3 is set to 0), the interrupt signal generated by the tilt function is pulsed: the duration of the pulse observed on the interrupt pins is about 60 µs; the duration of the pulse observed on the TILT_INT bit of the FUNC_CK_GATE register is 1/25 Hz.

If latch mode is enabled (LIR bit of CTRL3 is set to 1) and the interrupt signal is driven to the interrupt pins, once a tilt is detected, a reading of the FUNC_CK_GATE register clears the request on the INT2 pin and the TILT_INT bit of FUNC_CK_GATE register, and the device is ready to recognize the next tilt event. If latch mode is enabled but the interrupt signal is not driven to the interrupt pins, the TILT_INT bit of the FUNC_CK_GATE register is pulsed, with a fixed duration of 1/25 Hz.

The tilt function works at 25 Hz independently of the device ODR.

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

1. Write 20h in CTRL1 // Turn on the accelerometer // ODR = 25 Hz, FS = 2 *g*
2. Write 10h in FUNC_CTRL // Enable tilt detection
3. Write 10h in CTRL5 // Tilt detector interrupt driven to INT2 pin

5 First-in first-out (FIFO) buffer

In order to limit intervention by the host processor and facilitate post-processing data for events recognition, the LIS2DG embeds a first-in, first-out buffer (FIFO) for each of the three output channels, X, Y, and Z.

FIFO use allows consistent power saving for the system, it can wake up only when needed and burst the significant data out from the FIFO.

The FIFO buffer can work according to five different modes that guarantee a high level of flexibility during application development: Bypass mode, FIFO mode, Continuous mode, Bypass-to-Continuous and Continuous-to-FIFO mode.

A programmable watermark level and the FIFO_FULL event can be enabled to generate dedicated interrupts on the INT1 pin.

5.1 FIFO description

The FIFO buffer is able to store up to 256 acceleration samples of 14 bits for each channel or store the output of the acceleration module computation up to 768 entries (see); data are stored in the 14-bit 2's complement left-justified representation, which means that they always have to be right-shifted by two.

The data sample set consists of 6 bytes (Xl, Xh, Yl, Yh, Zl, and Zh) and they are released to the FIFO at the selected output data rate (ODR).

The new sample set is placed in the first empty FIFO slot until the buffer is full, therefore, the oldest value is overwritten.

Table 12: FIFO buffer full representation (256th sample set stored)

Output registers	28h	29h	2Ah	2Bh	2Ch	2Dh
	Xl	Xh	Yl	Yh	Zl	Zh
FIFO index	FIFO sample set					
FIFO(0)	Xl(0)	Xh(0)	Yl(0)	Yh(0)	Zl(0)	Zh(0)
FIFO(1)	Xl(1)	Xh(1)	Yl(1)	Yh(1)	Zl(1)	Zh(1)
FIFO(2)	Xl(2)	Xh(2)	Yl(2)	Yh(2)	Zl(2)	Zh(2)
FIFO(3)	Xl(3)	Xh(3)	Yl(3)	Yh(3)	Zl(3)	Zh(3)
...	...	...	...	...	...	...
FIFO(254)	Xl(254)	Xh(254)	Yl(254)	Yh(254)	Zl(254)	Zh(254)
FIFO(255)	Xl(255)	Xh(255)	Yl(255)	Yh(255)	Zl(255)	Zh(255)

Table 13: FIFO buffer full representation (257th sample set stored and 1st sample discarded)

Output registers	28h	29h	2Ah	2Bh	2Ch	2Dh
	Xl	Xh	Yl	Yh	Zl	Zh
FIFO index	**Sample set**					
FIFO(0)	Xl(1)	Xh(1)	Yl(1)	Yh(1)	Zl(1)	Zh(1)
FIFO(1)	Xl(2)	Xh(2)	Yl(2)	Yh(2)	Zl(2)	Zh(2)
FIFO(2)	Xl(3)	Xh(3)	Yl(3)	Yh(3)	Zl(3)	Zh(3)
FIFO(3)	Xl(4)	Xh(4)	Yl(4)	Yh(4)	Zl(4)	Zh(4)
...	...	...	...	...	...	...
FIFO(255)	Xl(256)	Xh(256)	Yl(256)	Yh(256)	Zl(256)	Zh(256)

represents the FIFO full status when 256 samples are stored in the buffer while represents the next step when the 257th sample is inserted into FIFO and the 1st sample is overwritten. The new oldest sample set is made available in the output registers.

When FIFO is enabled and the mode is different from Bypass, the LIS2DG output registers (28h to 2Dh) always contain the oldest FIFO sample set.

5.2 FIFO registers

The FIFO buffer is managed by four different accelerometer registers, two of these allow enabling and configuring the FIFO behavior, the other two provide information about the buffer status.

A few other registers are used to route FIFO events on the pad to interrupt the application processor. These are discussed in *Section 5.3: "FIFO interrupts"*.

5.2.1 FIFO_CTRL register (25h)

The FIFO_CTRL register contains the mode at which the FIFO is set. At reset by default the FIFO mode is Bypass which means off; the FIFO gets enabled and starts storing the samples (or the module) as soon as the mode is set to a mode other than Bypass.

Table 14: FIFO_CTRL register

b7	b6	b5	b4	b3	b2	b1	b0
FMODE2	FMODE1	FMODE0	-	MODULE_ TO_FIFO	-	-	IF_CS_ PU_DIS

The FMODE[2:0] bits select the FIFO buffer behavior:

1. FMODE[2:0] = 000b: Bypass mode (FIFO turned off)
2. FMODE[2:0] = 001b: FIFO mode
3. FMODE[2:0] = 011b: Continuous-to-FIFO mode
4. FMODE[2:0] = 100b: Bypass-to-continuous mode
5. FMODE[2:0] = 110b: Continuous mode

MODULE_TO_FIFO enables the module of the X/Y/Z samples (sqrt(x^2+y^2+z^2)) to go in FIFO instead of the samples themselves. Please note that the MODULE_ON bit of the FUNC_CTRL register must be on.

5.2.3 FIFO_THS register (2Eh)

This register may be used to set the FIFO threshold level.

Table 15: FIFO_THS register

b7	b6	b5	b4	b3	b2	b1	b0
FTH7	FTH6	FTH5	FTH4	FTH3	FTH2	FTH1	FTH0

The FTH[7:0] bits are intended to define the watermark level; when FIFO content is greater than or equal to this value, the FTH bit is set to “1” in the FIFO_SRC register.

5.2.4 FIFO_SRC (2Fh)

This register is updated at every ODR and provides information about the FIFO buffer status.

Table 16: FIFO_SRC register

b7	b6	b5	b4	b3	b2	b1	b0
FTH	FIFO_OVER _RUN	DIFF8	-	-	-	-	-

- FTH bit is set high when FIFO content exceeds watermark level. This flag can be routed to the pad (see *Section 5.3: "FIFO interrupts"*).
- FIFO_OVER_RUN bit is set high when the first sample is overwritten after the FIFO buffer is full. This means that the FIFO buffer contains 256 unread samples. The FIFO_OVER_RUN bit is reset when the first sample set has been read.
- DIFF8 bit (or FIFO_FULL bit) is used together with bits of FIFO_SAMPLES (DIFF[8:0]) to provide information of how many FIFO entries are used (000000000b means FIFO empty, 100000000b means FIFO full). This flag can be routed to the pad (see *Section 5.3: "FIFO interrupts"*).

The register content is updated synchronous to the FIFO write and read operation.

Table 17: FIFO_SRC behavior assuming FTH[7:0] = 15

FTH	DIFF8 (FIFO_FULL)	FIFO_OVER _RUN	DIFF[8:0]	Unread FIFO samples	Timing
0	0	0	000000000	0	t0
0	0	0	000000001	1	t0 + 1/ODR
0	0	0	000000010	2	t0 + 2/ODR
...	...	...	...	...	...
0	0	0	000001110	14	t0 + 14/ODR
1	0	0	000001111	15	t0 + 15/ODR
...	...	...	...	...	...
1	0	0	011111111	255	t0 + 255/ODR
1	1	0	100000000	256	t0 + 256/ODR
1	1	1	100000000	256	t0 + 257/ODR

5.2.5 FIFO_SAMPLES (30h)

The content of this register is used together with the DIFF8 bit of the FIFO_SRC register (DIFF[8:0]) to provide information of how many FIFO entries are used (000000000b means FIFO empty, 100000000b means FIFO full).

Table 18: FIFO_SAMPLES register

b7	b6	b5	b4	b3	b2	b1	b0
DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0

5.3 FIFO interrupts

There are two specific FIFO events that can be routed to the pad in order to interrupt the main processor: FIFO threshold and FIFO full.

The third FIFO event, FIFO_OVER_RUN, cannot be routed on the pad, but can instead be polled by reading the corresponding bit in the FIFO_SRC register.

5.3.1 FIFO threshold

The FIFO threshold is a configurable feature that can be used to generate a specific interrupt in order to know when the FIFO buffer contains at least the number of samples defined as the threshold level. The user can select the desired level in a range from 0 to 255 using the FTH[7:0] field in the FIFO_THS register.

If the number of entries in FIFO (DIFF[8:0]) is greater than or equal to the value programmed in FTH[7:0], the FTH bit is set high in the FIFO_SRC register.

DIFF[8:0] increases by one step at the ODR frequency and decreases by one step every time that a sample set reading is performed by the user.

The threshold flag (FTH) can be routed to the INT1 and INT2 pin to provide a dedicated interrupt for the application processor that can consume less power between each interrupt. The INT1_FTH bit of CTRL4 register and the INT2_FTH bit of CTRL5 register are dedicated to this task.

5.3.2 FIFO FULL

It is possible to configure the device to generate an interrupt whenever the FIFO gets full. To do so just set the INT1_FIFO_FULL bit of WAKE_UP_DUR register to ‘1’. To avoid losing samples, the FIFO reading operation must start and complete inside 1 ODR window.

5.4 Module-to-FIFO

If the module computation is on (MODULE_ON bit of FUNC_CTRL register is ‘1’) and the MODULE_TO_FIFO bit of FIFO_CTRL is set to ‘1’, then the FIFO buffer will not contain the acceleration samples but instead the computation of their module ($sqrt(x^2+y^2+z^2)$).

Since the module is a number of 14-bit in size, the FIFO buffer may contain up to 768 module entries (256 * 3).

The following is a simple procedure to enable the FIFO to contain the acceleration sample module:

1. Set MODULE_ON to 1 in FUNC_CTRL (3Fh) to enable module computation
2. Set MODULE_TO_FIFO bit of FIFO_CTRL (25h) to ‘1’ to save module in FIFO buffer
3. Enable the FIFO in one of the operative modes (see *Section 5.5: "FIFO modes"*)

5.5 FIFO modes

The LIS2DG FIFO buffer can be configured to operate in five different modes selectable by the FMODE[2:0] field in FIFO_CTRL register. Available configurations ensure a high-level of flexibility and extend the number of functions usable in application development.

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

5.5.1 Bypass mode

When Bypass mode is enabled, the FIFO is not operational: buffer content is cleared, output registers (0x28 to 0x2D) are frozen at the last value loaded, and the FIFO buffer remains empty until another mode is selected.

Bypass mode is activated by setting the FMODE[2:0] field to 000b in the FIFO_CTRL register.

Bypass mode must be used in order to stop and reset the FIFO buffer when a different mode is operating. Note that placing the FIFO buffer into Bypass mode clears the whole buffer content.

5.5.2 FIFO mode

In FIFO mode, the buffer continues filling until full (256 sample set stored). As soon as the FIFO_OVER_RUN flag gets 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 FMODE[2:0] field to 001b in the FIFO_CTRL register.

By selecting this mode, FIFO starts data collection and DIFF[8:0] changes according to the number of samples stored. At the end of the procedure, the FIFO_OVER_RUN flag rises to 1, and data can then be retrieved, performing a 256 sample set reading from the 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 exit Bypass mode.

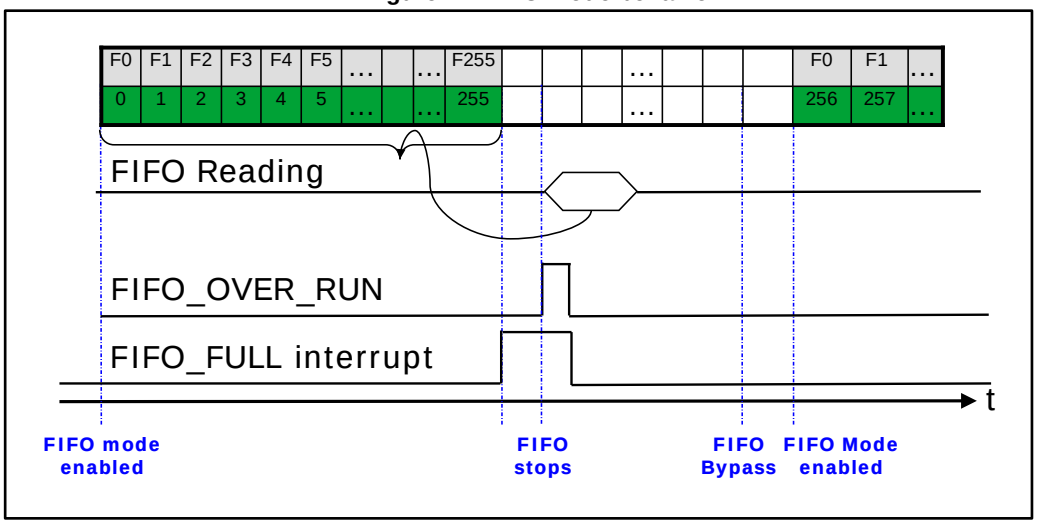
In order to serve the FIFO_FULL event as soon as possible, it is recommended to route it to the pad in order to generate an interrupt, which will then be managed by a specific handler:

1. Set INT1_FIFO_FULL to ‘1’: Enables FIFO_FULL interrupt
2. Set FMODE[2:0] = 001b: Enables FIFO mode

When the FIFO_FULL interrupt is generated:

1. Wait for the FIFO_OVER_RUN event (polling mode)
2. Read data from the accelerometer output registers

Figure 12: FIFO mode behavior



As indicated in *Figure 12: "FIFO mode behavior"*, when FIFO mode is enabled, the buffer starts to collect data and fills all the 256 slots (from F0 to F255) at the selected output data rate. When the buffer is full, as the next sample comes in and overrides the buffer, the FIFO_OVER_RUN 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 reading procedure may be performed inside an interrupt handler triggered by a FIFO_FULL condition (DIFF8) and it is composed of a 256 sample set of 6 bytes for a total of 1236 bytes and retrieves data starting from the oldest sample stored in FIFO (F0). The FIFO_OVER_RUN bit is reset when the first sample set has been read. The Bypass mode setting resets FIFO and allows the user to enable FIFO mode again.

5.5.3 Continuous mode

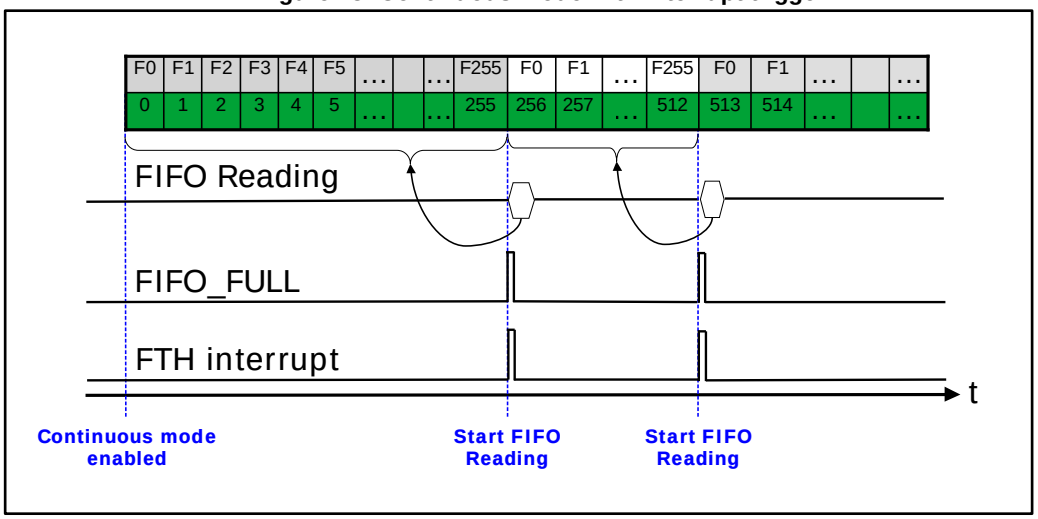
In Continuous mode FIFO continues filling, when the buffer is full, the FIFO index restarts from the beginning and older data is replaced by current data. The oldest values continue to be overwritten until a read operation frees FIFO slots. The host processor reading speed is most important in order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

Follow these steps for FIFO Continuous configuration which sets a threshold to generate an interrupt to trigger a read by the application processor:

1. Set FTH[7:0] to 255.
2. Set INT2_FTH to ‘1’: Enable FIFO threshold interrupt
3. Activate Continuous mode by setting the FMODE[2:0] field to 110b in the FIFO_CTRL register (25h).

When the FTH interrupt is generated, data is read from the accelerometer output registers.

Figure 13: Continuous mode with interrupt trigger



As indicated in *Figure 13: "Continuous mode with interrupt trigger"*, when Continuous mode is enabled, the FIFO buffer is continuously filling (from F0 to F255) at the selected output data rate. When the buffer is full, the FTH interrupt (as well as the FIFO_FULL flag, which might also be used to trigger an interrupt) goes high, and the application processor may read all FIFO samples (256 * 6 bytes) as soon as possible to avoid loss of data and to limit intervention by the host processor which increases system efficiency. See *Section 5.6: "Retrieving data from FIFO"* for more details on FIFO reading speed.

When a read command is sent to the device, the content of the output registers is moved to the SPI/I^2C register and the current oldest FIFO value is shifted into the output registers in order to allow the next read operation.

5.5.4 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 (e.g. wakeup, free-fall, tap, ...) occurs.

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

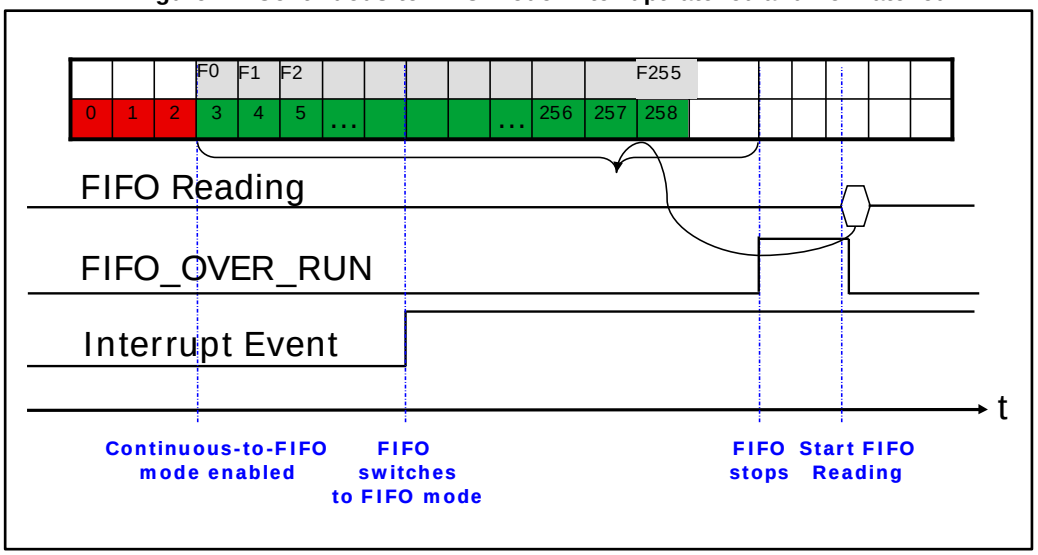
Follow these steps for Continuous-to-FIFO mode configuration:

1. Configure the desired interrupt generator by following the instructions in *Section 4: "Interrupt generation and Android embedded functions"* (be sure it is latched).
2. Activate Continuous-to-FIFO mode by setting the FMODE[2:0] field to 011b in the FIFO_CTRL register (25h).

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 or INT2 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 FIFO_OVER_RUN bit is set high and the next samples overwrite the oldest and the FIFO stops collecting data (see *Figure 14: "Continuous-to-FIFO mode: interrupt latched and non-latched"*).

Figure 14: Continuous-to-FIFO mode: interrupt latched and non-latched



5.5.5 Bypass-to-Continuous mode

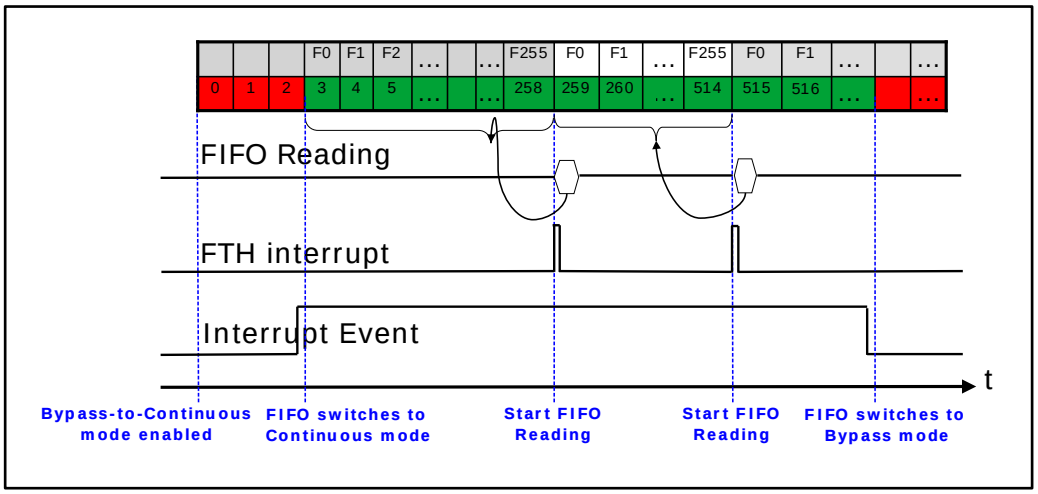
This mode is a combination of the Bypass and Continuous modes previously described. In Bypass-to-Continuous mode, the FIFO buffer starts in Bypass mode and switches to Continuous mode when the selected interrupt (e.g. wakeup, free-fall, tap, ...) occurs.

Follow these steps for Bypass-to-Continuous mode configuration:

1. Configure desired interrupt generator by following the instructions in *Section 4: "Interrupt generation and Android embedded functions"* (be sure it is latched).
2. Set FTH[7:0] to 255.
3. Set INT2_FTH to '1': Enables FIFO threshold interrupt
4. Activate Bypass-to-Continuous mode by setting the FMODE[2:0] field to 100b in the FIFO_CTRL register (25h).

When the FTH interrupt is generated, data is read from the accelerometer output registers.

Figure 15: Bypass-to-Continuous mode



As indicated in *Figure 15: "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 (e.g. a wakeup 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 FTH interrupt goes high, and the application processor may start reading all FIFO samples (256 * 6 bytes) as soon as possible to avoid loss of data.

If the FIFO_OVER_RUN flag was set, it will go 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.

Finally, either the interrupt event is cleared or the FIFO enters directly Bypass mode and then it stops collecting data.

5.6 Retrieving data from FIFO

When the FIFO mode is different from Bypass, reading the output registers (28h to 2Dh) returns the oldest FIFO sample set.

Whenever the output registers are read, their content is moved to the SPI/I^2C output buffer. FIFO slots are ideally shifted up one level in order to release room for receiving a new sample and the output registers load the current oldest value stored in the FIFO buffer.

The whole FIFO content is retrieved by performing 256 read operations from the accelerometer output registers; every other reading operation returns the same last value until a new sample set is available in the FIFO buffer.

Data can be retrieved from FIFO using every reading byte combination in order to increase application flexibility (ex: 1536 single byte read, 256 reads of 6 bytes, 1 multiple read of 1536 bytes, etc.).

It is recommended to read all FIFO slots in a multiple byte reading of 1536 bytes (6 output registers by 256 slots). In order to minimize communication between the master and slave the reading address may be automatically incremented by the device by setting the IF_ADD_INC bit of CTRL2 register to ‘1’; the device rolls back to 0x28 when register 0x2D is reached.

The I^2C speed is lower than SPI and it needs about 29 clock pulses to start communication (Start, Slave Address, Register Address+Write, Restart, Register Address+Read) plus an additional 9 clock pulses for every byte to read (total of 83 clock pulses). So, in the case of standard I^2C mode being used (max rate 100 kHz), a single sample set reading takes

830 µs while total FIFO download takes about 138.53 ms (29 + 9 * 1536 clock pulses).

In the case of the SPI, instead, 9 clock pulses are required only once at the very beginning to get started (r/w + Register Address) plus additional 8 clock pulses for every byte to read. With a 2 MHz clock a single sample set reading would take 28.5 µs, while total FIFO download takes about 6.15 ms.

If this recommendation were followed, using a standard I^2C (100 kHz) the complete FIFO reading (138.53 ms) is taking 14*ODR with ODR at 100 Hz. Using a SPI @2 MHz (10 MHz is the maximum supported by the device) the complete FIFO reading would take 1*ODR with ODR at 100 Hz.

So, in order to not lose samples, the application will read samples before the FIFO becomes full, setting a threshold and using the FTH interrupt (see section *Section 5.3: "FIFO interrupts"*).

Table 19: Example: threshold function of ODR

ODR (Hz)	FTH_THS (I^2C @ 100 kHz)	FTH_THS (I^2C @ 400 kHz)	FTH_THS (SPI @ 2 MHz
50	7	2	0
100	14	4	1
200	28	7	2
400	56	14	3
800	112	28	5
1600	224	56	10

6 Temperature sensor

The LIS2DG is provided with an internal temperature sensor that is suitable for ambient temperature measurement.

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

The output data rate of temperature sensor is fixed at 12.5 Hz.

The temperature data is given by the OUT_T register and it is represented as a number of

8 bits in two's complement format, with a sensitivity of +1 LSB/°C. The output zero level corresponds to 25 °C.

6.1 Example of temperature data calculation

Table 20: "Output data registers content vs. temperature" provides a few basic examples of the data that is read in the temperature data registers at different ambient temperature values. The values listed in this table are given under the hypothesis of perfect device calibration (i.e. no offset, no gain error,....).

Table 20: Output data registers content vs. temperature

Temperature values	OUT_T (26h)
23 °C	FEh
24 °C	FFh
25 °C	00h
27 °C	02h

7 Self-test

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

7.1 Accelerometer self-test

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

The accelerometer self-test function is off when the ST[2:1] bits of the CTRL3 register are programmed to 00b; it is enabled when the ST[2:1] bits are set to 01b (positive sign self-test) or 10b (negative sign self-test).

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

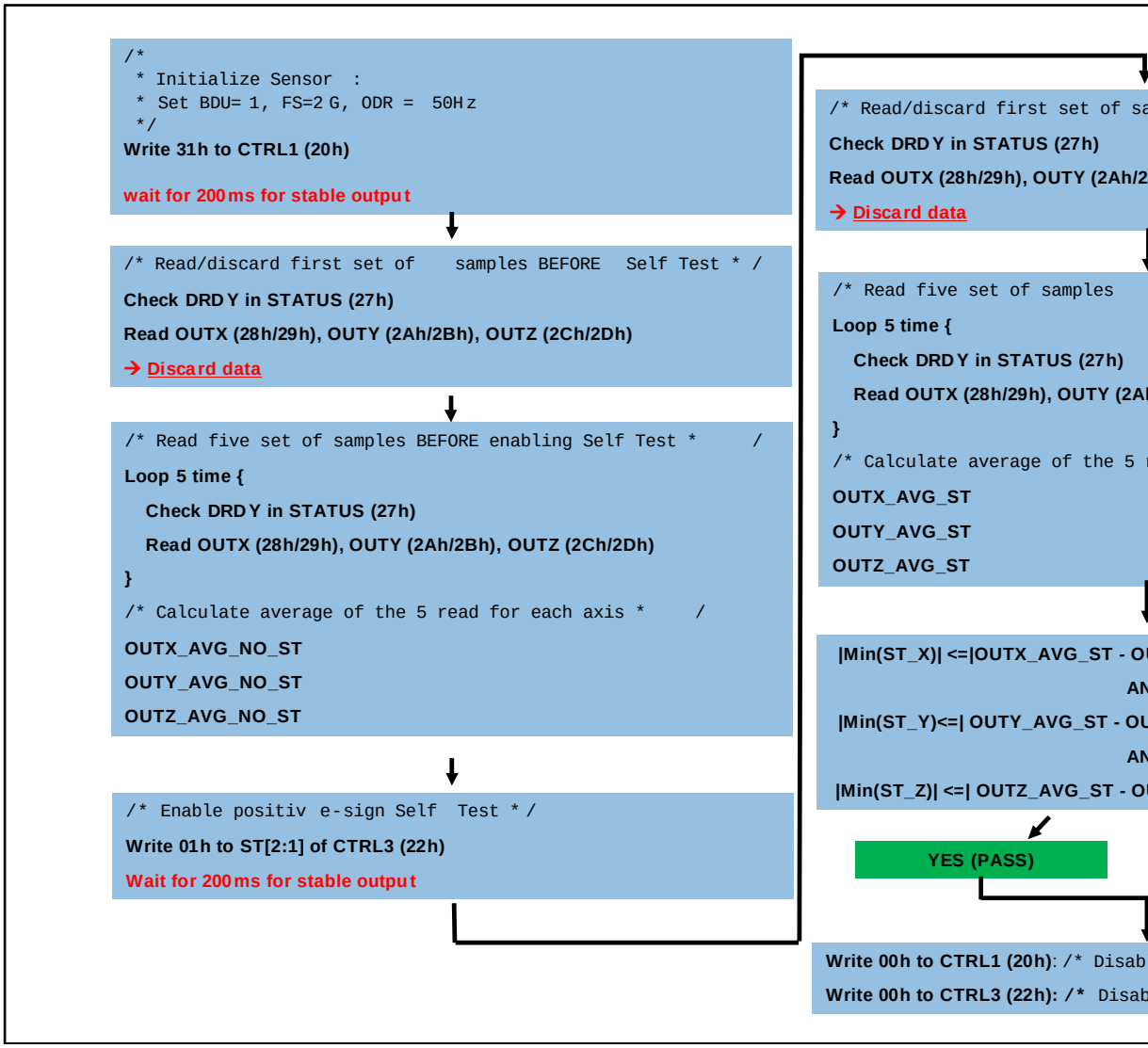
The procedure consists of

1. enabling the accelerometer
2. averaging five samples before enabling the self-test
3. averaging fives samples after enabling the self-test
4. computing the difference in module for each axis and verify it falls in a given range. The min and max value are provided in the datasheet.

The complete accelerometer self-test procedure is indicated in *Figure 16: "Accelerometer self-test procedure"*.

Note: Keep the device still during the self-test procedure.

Figure 16: Accelerometer self-test procedure



8 Revision history

Table 21: Document revision history

Date	Revision	Changes
08-Jul-2015	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.