# LSM6DSV80X: 6-axis IMU with high-*g* accelerometer, embedded AI, and sensor fusion for wearables and sport trackers

## Introduction

This document provides usage information and application hints related to ST's LSM6DSV80X 6-axis IMU (inertial measurement unit).

The LSM6DSV80X includes a 3-axis digital low-*g* accelerometer, 3-axis digital high-*g* accelerometer, and 3-axis digital gyroscope system-in-package with a digital I²C, SPI, and MIPI I3C® serial interface standard output. Thanks to the ultralow noise performance of both the gyroscope and the accelerometer, the device combines always-on low-power features with superior sensing precision for an optimal motion experience for the consumer, and "smart, always-aware" features for system power optimization. Furthermore, the low-*g* accelerometer features smart sleep-to-wake-up (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* for the low-*g* accelerometer, a range of ±32/±64/±80 *g* for the high-*g* accelerometer, and an angular rate range of ±250/±500/±1000/±2000/±4000 dps. It features the capability to enable up to three different cores for UI data processing.

The LSM6DSV80X 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.

Connection mode 2 implements the sensor hub functionality that allows connecting external sensors to the device.

The device 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 can be programmed independently for motion detection or gesture recognition such as glance, absolute wrist tilt, shake, double-shake, or pick-up.

The LSM6DSV80X also embeds machine learning core logic, which 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 device has an integrated smart first-in first-out (FIFO) buffer of up to 4.5 KB size, allowing dynamic batching of significant data (that is, external sensors, step counter, timestamp and temperature, SFLP-generated data, and MLC exported filters and features).

The LSM6DSV80X is available in a small plastic, land grid array package (LGA-14L) 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 smartphones, IoT connected devices, and wearables or any other application where reduced package size and weight are required.

# 1 Pin description

Figure 1. Pin connections

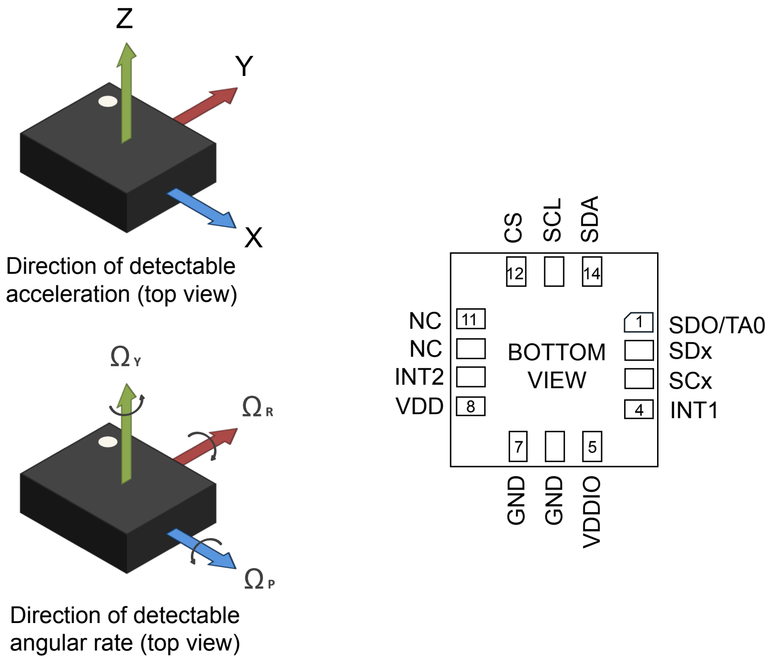


**Table 1. Internal pin status**

| Pin# | Name | Mode 1 function | Mode 2 function | Pin status mode 1 | Pin status mode 2 |
|---|---|---|---|---|---|
| 1 | SDO | SPI 4-wire interface serial data output (SDO) | SPI 4-wire interface serial data output (SDO) | Default: input without pull-up<br>Pull-up is enabled if bit SDO_PU_EN = 1 in register PIN_CTRL (02h). | |
| | TA0 | I²C least significant bit of the device address (TA0)<br>MIPI I3C® least significant bit of the static address (TA0) | I²C least significant bit of the device address (TA0)<br>MIPI I3C® least significant bit of the static address (TA0) | | |
| 2 | SDx | Connect to VDDIO or GND. | I²C controller serial data (CSDA) | Default: input without pull-up<br>Pull-up is enabled if bit SHUB_PU_EN = 1 in register IF_CFG (03h). | |
| 3 | SCx | Connect to VDDIO or GND. | I²C controller serial clock (CSCL) | Default: input without pull-up<br>Pull-up is enabled if bit SHUB_PU_EN = 1 in register IF_CFG (03h). | |
| 4 | INT1 | Programmable interrupt 1 | Programmable interrupt 1 | Default: output forced to ground | |
| 5 | VDDIO | Power supply for I/O pins | Power supply for I/O pins | | |
| 6 | GND | 0 V supply | 0 V supply | | |
| 7 | GND | 0 V supply | 0 V supply | | |
| 8 | VDD | Power supply | Power supply | | |
| 9 | INT2 | Programmable interrupt 2 (INT2) | Programmable interrupt 2 (INT2) / I²C controller external synchronization signal (CDRDY) | Default: output forced to ground | |
| 10 | NC | Leave unconnected | Leave unconnected | | |
| 11 | NC | Leave unconnected | Leave unconnected | | |
| 12 | CS | I²C/SPI mode selection<br>(1: SPI idle mode / I²C communication enabled;<br>0: SPI communication mode / I²C disabled) | I²C/SPI mode selection<br>(1: SPI idle mode / I²C communication enabled;<br>0: SPI communication mode / I²C disabled) | Default: input with pull-up<br>Pull-up is disabled if bit I2C_I3C_disable = 1 in register IF_CFG (03h). | |
| 13 | SCL | I²C/MIPI I3C® serial clock (SCL) / SPI serial port clock (SPC) | I²C/MIPI I3C® serial clock (SCL) / SPI serial port clock (SPC) | Default: input without pull-up | |
| 14 | SDA | I²C/MIPI I3C® serial data (SDA) / SPI serial data input (SDI) / 3-wire interface serial data output (SDO) | I²C/MIPI I3C® serial data (SDA) / SPI serial data input (SDI) / 3-wire interface serial data output (SDO) | Default: input without pull-up<br>Pull-up is enabled if bit SDA_PU_EN = 1 in register IF_CFG (03h). | |

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

# 2 Registers

**Table 2. Registers**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| FUNC_CFG_ACCESS | 01h | EMB_FUNC_ REG_ACCESS | SHUB_REG _ACCESS | 0 | 0 | FSM_WR_ CTRL_EN | SW_POR | 0 | 0 |
| PIN_CTRL | 02h | 0 | SDO_PU_EN | IBHR_ POR_EN | 0 | 0 | 0 | IO_PAD_ STRENGTH_1 | IO_PAD_ STRENGTH_0 |
| IF_CFG | 03h | SDA_PU_EN | SHUB_PU_EN | ASF_CTRL | H_LACTIVE | PP_OD | SIM | 0 | I2C_I3C_ disable |
| ODR_TRIG_CFG | 06h | ODR_TRIG_ NODR_7 | ODR_TRIG_ NODR_6 | ODR_TRIG_ NODR_5 | ODR_TRIG_ NODR_4 | ODR_TRIG_ NODR_3 | ODR_TRIG_ NODR_2 | ODR_TRIG_ NODR_1 | ODR_TRIG_ NODR_0 |
| FIFO_CTRL1 | 07h | WTM_7 | WTM_6 | WTM_5 | WTM_4 | WTM_3 | WTM_2 | WTM_1 | WTM_0 |
| FIFO_CTRL2 | 08h | STOP_ON_WTM | FIFO_COMPR_ RT_EN | 0 | ODR_CHG_EN | 0 | UNCOMPR_ RATE_1 | UNCOMPR_ RATE_0 | 0 |
| FIFO_CTRL3 | 09h | BDR_GY_3 | BDR_GY_2 | BDR_GY_1 | BDR_GY_0 | BDR_XL_3 | BDR_XL_2 | BDR_XL_1 | BDR_XL_0 |
| FIFO_CTRL4 | 0Ah | DEC_TS_BATCH_1 | DEC_TS_BATCH_0 | ODR_T_BATCH_1 | ODR_T_BATCH_0 | 0 | FIFO_MODE_2 | FIFO_MODE_1 | FIFO_MODE_0 |
| COUNTER_BDR_REG1 | 0Bh | 0 | TRIG_COUNTER _BDR_1 | TRIG_COUNTER _BDR_0 | 0 | XL_HG_BATCH_EN | 0 | CNT_BDR_TH_9 | CNT_BDR_TH_8 |
| COUNTER_BDR_REG2 | 0Ch | CNT_BDR_TH_7 | CNT_BDR_TH_6 | CNT_BDR_TH_5 | CNT_BDR_TH_4 | CNT_BDR_TH_3 | CNT_BDR_TH_2 | CNT_BDR_TH_1 | CNT_BDR_TH_0 |
| INT1_CTRL | 0Dh | 0 | INT1_CNT_BDR | INT1_FIFO_FULL | INT1_FIFO_OVR | INT1_FIFO_TH | 0 | INT1_DRDY_G | INT1_DRDY_XL |
| INT2_CTRL | 0Eh | INT2_EMB_ FUNC_ENDOP | INT2_CNT_BDR | INT2_FIFO_FULL | INT2_FIFO_OVR | INT2_FIFO_TH | 0 | INT2_DRDY_G | INT2_DRDY_XL |
| WHO_AM_I | 0Fh | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| CTRL1 | 10h | 0 | OP_MODE_XL_2 | OP_MODE_XL_1 | OP_MODE_XL_0 | ODR_XL_3 | ODR_XL_2 | ODR_XL_1 | ODR_XL_0 |
| CTRL2 | 11h | 0 | OP_MODE_G_2 | OP_MODE_G_1 | OP_MODE_G_0 | ODR_G_3 | ODR_G_2 | ODR_G_1 | ODR_G_0 |
| CTRL3 | 12h | BOOT | BDU | 0 | 0 | 0 | IF_INC | 0 | SW_RESET |
| CTRL4 | 13h | 0 | 0 | 0 | INT2_on_INT1 | DRDY_MASK | INT2_DRDY_TEMP | DRDY_PULSED | INT2_IN_LH |
| CTRL5 | 14h | 0 | 0 | 0 | 0 | 0 | BUS_ACT_SEL_1 | BUS_ACT_SEL_0 | INT_EN_I3C |
| CTRL6 | 15h | 0 | LPF1_G_BW_2 | LPF1_G_BW_1 | LPF1_G_BW_0 | 1 | FS_G_2 | FS_G_1 | FS_G_0 |
| CTRL7 | 16h | INT1_DRDY_ XL_HG | INT2_DRDY_ XL_HG | 0 | 0 | 0 | 0 | 0 | LPF1_G_EN |
| CTRL8 | 17h | HP_LPF2_ XL_BW_2 | HP_LPF2_ XL_BW_1 | HP_LPF2_ XL_BW_0 | 0 | 0 | 0 | FS_XL_1 | FS_XL_0 |
| CTRL9 | 18h | 0 | HP_REF_ MODE_XL | XL_FASTSETTL_ MODE | HP_SLOPE_XL_EN | LPF2_XL_EN | 0 | USR_OFF_W | USR_OFF_ ON_OUT |
| CTRL10 | 19h | 0 | EMB_FUNC _DEBUG | 0 | 0 | ST_G_1 | ST_G_0 | ST_XL_1 | ST_XL_0 |
| CTRL_STATUS | 1Ah | 0 | 0 | 0 | 0 | 0 | FSM_WR_ CTRL_STATUS | - | 0 |
| FIFO_STATUS1 | 1Bh | DIFF_FIFO_7 | DIFF_FIFO_6 | DIFF_FIFO_5 | DIFF_FIFO_4 | DIFF_FIFO_3 | DIFF_FIFO_2 | DIFF_FIFO_1 | DIFF_FIFO_0 |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| FIFO_STATUS2 | 1Ch | FIFO_WTM_IA | FIFO_OVR_IA | FIFO_FULL_IA | COUNTER_BDR_IA | FIFO_OVR_ LATCHED | 0 | 0 | DIFF_FIFO_8 |
| ALL_INT_SRC | 1Dh | EMB_FUNC_IA | SHUB_IA | SLEEP_ CHANGE_IA | D6D_IA | HG_IA | TAP_IA | WU_IA | FF_IA |
| STATUS_REG | 1Eh | TIMESTAMP_ ENDCOUNT | 0 | 0 | 0 | XLHGDA | TDA | GDA | XLDA |
| OUT_TEMP_L | 20h | Temp7 | Temp6 | Temp5 | Temp4 | Temp3 | Temp2 | Temp1 | Temp0 |
| OUT_TEMP_H | 21h | Temp15 | Temp14 | Temp13 | Temp12 | Temp11 | Temp10 | Temp9 | Temp8 |
| OUTX_L_G | 22h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTX_H_G | 23h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| OUTY_L_G | 24h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTY_H_G | 25h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| OUTZ_L_G | 26h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTZ_H_G | 27h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| OUTX_L_A | 28h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTX_H_A | 29h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| OUTY_L_A | 2Ah | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTY_H_A | 2Bh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| OUTZ_L_A | 2Ch | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| OUTZ_H_A | 2Dh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| UI_OUTX_L_A_HG | 34h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UI_OUTX_H_A_HG | 35h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| UI_OUTY_L_A_HG | 36h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UI_OUTY_H_A_HG | 37h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| UI_OUTZ_L_A_HG | 38h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UI_OUTZ_H_A_HG | 39h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TIMESTAMP0 | 40h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TIMESTAMP1 | 41h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TIMESTAMP2 | 42h | D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 |
| TIMESTAMP3 | 43h | D31 | D30 | D29 | D28 | D27 | D26 | D25 | D24 |
| UI_STATUS_REG | 44h | 0 | 0 | 0 | 0 | 0 | GYRO_SETTLING | 0 | 0 |
| WAKE_UP_SRC | 45h | 0 | SLEEP_ CHANGE_IA | FF_IA | SLEEP_STATE | WU_IA | X_WU | Y_WU | Z_WU |
| TAP_SRC | 46h | 0 | TAP_IA | SINGLE_TAP | DOUBLE_TAP | TAP_SIGN | X_TAP | Y_TAP | Z_TAP |
| D6D_SRC | 47h | 0 | D6D_IA | ZH | ZL | YH | YL | XH | XL |
| STATUS_CONTROLLER_ MAINPAGE | 48h | WR_ONCE_DONE | TARGET3_NACK | TARGET2_NACK | TARGET1_NACK | TARGET0_NACK | 0 | 0 | SENS_HUB_ ENDOP |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| EMB_FUNC_STATUS_ MAINPAGE | 49h | IS_FSM_LC | 0 | IS_SIGMOT | IS_TILT | IS_STEP_DET | 0 | 0 | 0 |
| FSM_STATUS_MAINPAGE | 4Ah | IS_FSM8 | IS_FSM7 | IS_FSM6 | IS_FSM5 | IS_FSM4 | IS_FSM3 | IS_FSM2 | IS_FSM1 |
| MLC_STATUS_MAINPAGE | 4Bh | IS_MLC8 | IS_MLC7 | IS_MLC6 | IS_MLC5 | IS_MLC4 | IS_MLC3 | IS_MLC2 | IS_MLC1 |
| HG_WAKE_UP_SRC | 4Ch | - | HG_SHOCK_ CHANGE_IA | HG_SHOCK_ STATE | HG_WU_CHANGE_ IA | HG_WU_IA | HG_X_WU | HG_Y_WU | HG_Z_WU |
| CTRL2_XL_HG | 4Dh | 0 | 0 | 0 | HG_USR_ OFF_ON_WU | 0 | 0 | XL_HG_ST1 | XL_HG_ST0 |
| CTRL1_XL_HG | 4Eh | XL_HG_REGOUT_ EN | HG_USR_OFF_ON _OUT | ODR_XL_HG_2 | ODR_XL_HG_1 | ODR_XL_HG_0 | FS_XL_HG_2 | FS_XL_HG_1 | FS_XL_HG_0 |
| INTERNAL_FREQ_FINE | 4Fh | FREQ_FINE_7 | FREQ_FINE_6 | FREQ_FINE_5 | FREQ_FINE_4 | FREQ_FINE_3 | FREQ_FINE_2 | FREQ_FINE_1 | FREQ_FINE_0 |
| FUNCTIONS_ENABLE | 50h | INTERRUPTS_ ENABLE | TIMESTAMP_EN | 0 | 0 | DIS_RST_LIR_ ALL_INT | 0 | INACT_EN_1 | INACT_EN_0 |
| HG_FUNCTIONS_ENABLE | 52h | HG_INTERRUPTS_ ENABLE | HG_WU_CHANGE_ INT_SEL | INT2_HG_WU | INT1_HG_WU | HG_SHOCK_ DUR_3 | HG_SHOCK_ DUR_2 | HG_SHOCK_ DUR_1 | HG_SHOCK_ DUR_0 |
| HG_WAKE_UP_THS | 53h | HG_WK_THS_7 | HG_WK_THS_6 | HG_WK_THS_5 | HG_WK_THS_4 | HG_WK_THS_3 | HG_WK_THS_2 | HG_WK_THS_1 | HG_WK_THS_0 |
| INACTIVITY_DUR | 54h | SLEEP_STATUS_ ON_INT | WU_INACT_ THS_W_2 | WU_INACT_ THS_W_1 | WU_INACT_ THS_W_0 | XL_INACT_ODR_1 | XL_INACT_ODR_0 | INACT_DUR_1 | INACT_DUR_0 |
| INACTIVITY_THS | 55h | INT2_HG_SHOCK_ CHANGE | INT1_HG_SHOCK_ CHANGE | INACT_THS_5 | INACT_THS_4 | INACT_THS_3 | INACT_THS_2 | INACT_THS_1 | INACT_THS_0 |
| TAP_CFG0 | 56h | 0 | LOW_PASS_ ON_6D | HW_FUNC_MASK _XL_SETTL | SLOPE_FDS | TAP_X_EN | TAP_Y_EN | TAP_Z_EN | LIR |
| TAP_CFG1 | 57h | TAP_PRIORITY_2 | TAP_PRIORITY_1 | TAP_PRIORITY_0 | TAP_THS_X_4 | TAP_THS_X_3 | TAP_THS_X_2 | TAP_THS_X_1 | TAP_THS_X_0 |
| TAP_CFG2 | 58h | 0 | 0 | 0 | TAP_THS_Y_4 | TAP_THS_Y_3 | TAP_THS_Y_2 | TAP_THS_Y_1 | TAP_THS_Y_0 |
| TAP_THS_6D | 59h | D4D_EN | SIXD_THS_1 | SIXD_THS_0 | TAP_THS_Z_4 | TAP_THS_Z_3 | TAP_THS_Z_2 | TAP_THS_Z_1 | TAP_THS_Z_0 |
| TAP_DUR | 5Ah | DUR_3 | DUR_2 | DUR_1 | DUR_0 | QUIET_1 | QUIET_0 | SHOCK_1 | SHOCK_0 |
| WAKE_UP_THS | 5Bh | SINGLE_ DOUBLE_TAP | USR_OFF_ON_WU | WK_THS_5 | WK_THS_4 | WK_THS_3 | WK_THS_2 | WK_THS_1 | WK_THS_0 |
| WAKE_UP_DUR | 5Ch | FF_DUR_5 | WAKE_DUR_1 | WAKE_DUR_0 | 0 | SLEEP_DUR_3 | SLEEP_DUR_2 | SLEEP_DUR_1 | SLEEP_DUR_0 |
| FREE_FALL | 5Dh | FF_DUR_4 | FF_DUR_3 | FF_DUR_2 | FF_DUR_1 | FF_DUR_0 | FF_THS_2 | FF_THS_1 | FF_THS_0 |
| MD1_CFG | 5Eh | INT1_SLEEP _CHANGE | INT1_ SINGLE_TAP | INT1_WU | INT1_FF | INT1_ DOUBLE_TAP | INT1_6D | INT1_EMB_FUNC | INT1_SHUB |
| MD2_CFG | 5Fh | INT2_SLEEP _CHANGE | INT2_ SINGLE_TAP | INT2_WU | INT2_FF | INT2_ DOUBLE_TAP | INT2_6D | INT2_EMB_FUNC | INT2_TIMESTAMP |
| HAODR_CFG | 62h | 0 | 0 | 0 | 0 | 0 | 0 | HAODR_SEL_1 | HAODR_SEL_0 |
| EMB_FUNC_CFG | 63h | HG_USR_OFF_ON _EMB_FUNC | EMB_FUNC_IRQ _MASK_XL_HG_SE TTL | EMB_FUNC_IRQ _MASK_G_SETTL | EMB_FUNC_IRQ _MASK_XL_SETTL | EMB_FUNC_ DISABLE | 0 | 0 | 0 |
| XL_HG_X_OFS_USR | 6Ch | XL_HG_X_OFS_ USR_7 | XL_HG_X_OFS_ USR_6 | XL_HG_X_OFS_ USR_5 | XL_HG_X_OFS_ USR_4 | XL_HG_X_OFS_ USR_3 | XL_HG_X_OFS_ USR_2 | XL_HG_X_OFS_ USR_1 | XL_HG_X_OFS_ USR_0 |
| XL_HG_Y_OFS_USR | 6Dh | XL_HG_Y_OFS_ USR_7 | XL_HG_Y_OFS_ USR_6 | XL_HG_Y_OFS_ USR_5 | XL_HG_Y_OFS_ USR_4 | XL_HG_Y_OFS_ USR_3 | XL_HG_Y_OFS_ USR_2 | XL_HG_Y_OFS_ USR_1 | XL_HG_Y_OFS_ USR_0 |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| XL_HG_Z_OFS_USR | 6Eh | XL_HG_Z_OFS_ USR_7 | XL_HG_Z_OFS_ USR_6 | XL_HG_Z_OFS_ USR_5 | XL_HG_Z_OFS_ USR_4 | XL_HG_Z_OFS_ USR_3 | XL_HG_Z_OFS_ USR_2 | XL_HG_Z_OFS_ USR_1 | XL_HG_Z_OFS_ USR_0 |
| X_OFS_USR | 73h | X_OFS_USR_7 | X_OFS_USR_6 | X_OFS_USR_5 | X_OFS_USR_4 | X_OFS_USR_3 | X_OFS_USR_2 | X_OFS_USR_1 | X_OFS_USR_0 |
| Y_OFS_USR | 74h | Y_OFS_USR_7 | Y_OFS_USR_6 | Y_OFS_USR_5 | Y_OFS_USR_4 | Y_OFS_USR_3 | Y_OFS_USR_2 | Y_OFS_USR_1 | Y_OFS_USR_0 |
| Z_OFS_USR | 75h | Z_OFS_USR_7 | Z_OFS_USR_6 | Z_OFS_USR_5 | Z_OFS_USR_4 | Z_OFS_USR_3 | Z_OFS_USR_2 | Z_OFS_USR_1 | Z_OFS_USR_0 |
| FIFO_DATA_OUT_TAG | 78h | TAG_SENSOR_4 | TAG_SENSOR_3 | TAG_SENSOR_2 | TAG_SENSOR_1 | TAG_SENSOR_0 | TAG_CNT_1 | TAG_CNT_0 | - |
| FIFO_DATA_OUT_X_L | 79h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| FIFO_DATA_OUT_X_H | 7Ah | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| FIFO_DATA_OUT_Y_L | 7Bh | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| FIFO_DATA_OUT_Y_H | 7Ch | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| FIFO_DATA_OUT_Z_L | 7Dh | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| FIFO_DATA_OUT_Z_H | 7Eh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |

## 2.1 Embedded functions registers

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

**Table 3. Embedded functions registers**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| PAGE_SEL | 02h | PAGE_SEL3 | PAGE_SEL2 | PAGE_SEL1 | PAGE_SEL0 | 0 | 0 | 0 | 1 |
| EMB_FUNC_EN_A | 04h | MLC_BEFORE _FSM_EN | 0 | SIGN_MOTION_EN | TILT_EN | PEDO_EN | 0 | SFLP_GAME_EN | 0 |
| EMB_FUNC_EN_B | 05h | 0 | 0 | 0 | MLC_EN | FIFO_COMPR_EN | 0 | 0 | FSM_EN |
| EMB_FUNC_EXEC_STATUS | 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 | INT1_SIG_MOT | INT1_TILT | INT1_STEP_ DETECTOR | 0 | 0 | 0 |
| FSM_INT1 | 0Bh | INT1_FSM8 | INT1_FSM7 | INT1_FSM6 | INT1_FSM5 | INT1_FSM4 | INT1_FSM3 | INT1_FSM2 | INT1_FSM1 |
| MLC_INT1 | 0Dh | INT1_MLC8 | INT1_MLC7 | INT1_MLC6 | INT1_MLC5 | INT1_MLC4 | INT1_MLC3 | INT1_MLC2 | INT1_MLC1 |
| EMB_FUNC_INT2 | 0Eh | INT2_FSM_LC | 0 | INT2_SIG_MOT | INT2_TILT | INT2_STEP_ DETECTOR | 0 | 0 | 0 |
| FSM_INT2 | 0Fh | INT2_FSM8 | INT2_FSM7 | INT2_FSM6 | INT2_FSM5 | INT2_FSM4 | INT2_FSM3 | INT2_FSM2 | INT2_FSM1 |
| MLC_INT2 | 11h | INT2_MLC8 | INT2_MLC7 | INT2_MLC6 | INT2_MLC5 | INT2_MLC4 | INT2_MLC3 | INT2_MLC2 | INT2_MLC1 |
| EMB_FUNC_STATUS | 12h | IS_FSM_LC | 0 | IS_SIGMOT | IS_TILT | IS_STEP_DET | 0 | 0 | 0 |
| FSM_STATUS | 13h | IS_FSM8 | IS_FSM7 | IS_FSM6 | IS_FSM5 | IS_FSM4 | IS_FSM3 | IS_FSM2 | IS_FSM1 |
| MLC_STATUS | 15h | IS_MLC8 | IS_MLC7 | IS_MLC6 | IS_MLC5 | IS_MLC4 | IS_MLC3 | IS_MLC2 | IS_MLC1 |
| PAGE_RW | 17h | EMB_FUNC_LIR | PAGE_WRITE | PAGE_READ | 0 | 0 | 0 | 0 | 0 |
| SFLP_GBIASX_L | 18h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASX_H | 19h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GBIASY_L | 1Ah | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASY_H | 1Bh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GBIASZ_L | 1Ch | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASZ_H | 1Dh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GRAVX_L | 1Eh | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GRAVX_H | 1Fh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GRAVY_L | 20h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GRAVY_H | 21h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GRAVZ_L | 22h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GRAVZ_H | 23h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_QUATW_L | 2Ah | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_QUATW_H | 2Bh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| SFLP_QUATX_L | 2Ch | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_QUATX_H | 2Dh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_QUATY_L | 2Eh | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_QUATY_H | 2Fh | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_QUATZ_L | 30h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_QUATZ_H | 31h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GBIASX_INIT_L | 32h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASX_INIT_H | 33h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GBIASY_INIT_L | 34h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASY_INIT_H | 35h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| SFLP_GBIASZ_INIT_L | 36h | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SFLP_GBIASZ_INIT_H | 37h | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| EMB_FUNC_FIFO_EN_A | 44h | MLC_FIFO_EN | STEP_COUNTER _FIFO_EN | SFLP_GBIAS _FIFO_EN | SFLP_GRAVITY _FIFO_EN | 0 | 0 | SFLP_GAME _FIFO_EN | 0 |
| EMB_FUNC_FIFO_EN_B | 45h | 0 | 0 | 0 | 0 | 0 | FSM_FIFO_EN | MLC_FILTER_ FEATURE_FIFO_EN | 0 |
| FSM_ENABLE | 46h | FSM8_EN | FSM7_EN | FSM6_EN | FSM5_EN | FSM4_EN | FSM3_EN | FSM2_EN | FSM1_EN |
| FSM_LONG_COUNTER_L | 48h | FSM_LC_7 | FSM_LC_6 | FSM_LC_5 | FSM_LC_4 | FSM_LC_3 | FSM_LC_2 | FSM_LC_1 | FSM_LC_0 |
| FSM_LONG_COUNTER_H | 49h | FSM_LC_15 | FSM_LC_14 | FSM_LC_13 | FSM_LC_12 | FSM_LC_11 | FSM_LC_10 | FSM_LC_9 | FSM_LC_8 |
| INT_ACK_MASK | 4Bh | IACK_MASK7 | IACK_MASK6 | IACK_MASK5 | IACK_MASK4 | IACK_MASK3 | IACK_MASK2 | IACK_MASK1 | IACK_MASK0 |
| FSM_OUTS1 | 4Ch | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS2 | 4Dh | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS3 | 4Eh | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS4 | 4Fh | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS5 | 50h | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS6 | 51h | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS7 | 52h | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| FSM_OUTS8 | 53h | P_X | N_X | P_Y | N_Y | P_Z | N_Z | P_V | N_V |
| SFLP_ODR | 5Eh | 0 | 1 | SFLP_GAME _ODR_2 | SFLP_GAME _ODR_1 | SFLP_GAME _ODR_0 | 0 | 1 | 1 |
| FSM_ODR | 5Fh | 0 | 1 | FSM_ODR_2 | FSM_ODR_1 | FSM_ODR_0 | 0 | 1 | 1 |
| MLC_ODR | 60h | 0 | MLC_ODR_2 | MLC_ODR_1 | MLC_ODR_0 | 0 | 1 | 0 | 1 |
| STEP_COUNTER_L | 62h | STEP_7 | STEP_6 | STEP_5 | STEP_4 | STEP_3 | STEP_2 | STEP_1 | STEP_0 |
| STEP_COUNTER_H | 63h | STEP_15 | STEP_14 | STEP_13 | STEP_12 | STEP_11 | STEP_10 | STEP_9 | STEP_8 |
| EMB_FUNC_SRC | 64h | PEDO_RST_STEP | 0 | STEP_DETECTED | STEP_COUNT_ DELTA_IA | STEP_OVERFLOW | STEPCOUNTER _BIT_SET | 0 | 0 |
| EMB_FUNC_INIT_A | 66h | MLC_BEFORE _FSM_INIT | 0 | SIG_MOT_INIT | TILT_INIT | STEP_DET_INIT | 0 | SFLP_GAME _INIT | 0 |
| EMB_FUNC_INIT_B | 67h | 0 | 0 | 0 | MLC_INIT | FIFO_COMPR _INIT | PT_INIT | 0 | FSM_INIT |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| EMB_FUNC_SENSOR_CONV_EN | 6Eh | 0 | 0 | 0 | 0 | EXT_SENSOR_CONV_EN | TEMP_CONV_EN | GYRO_CONV_EN | XL_HG_CONV_EN |
| MLC1_SRC | 70h | 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 | 71h | 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 | 72h | 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 | 73h | MLC4_SRC_7 | MLC4_SRC_6 | MLC4_SRC_5 | MLC4_SRC_4 | MLC4_SRC_3 | MLC4_SRC_2 | MLC4_SRC_1 | MLC4_SRC_0 |
| MLC5_SRC | 74h | MLC5_SRC_7 | MLC5_SRC_6 | MLC5_SRC_5 | MLC5_SRC_4 | MLC5_SRC_3 | MLC5_SRC_2 | MLC5_SRC_1 | MLC5_SRC_0 |
| MLC6_SRC | 75h | MLC6_SRC_7 | MLC6_SRC_6 | MLC6_SRC_5 | MLC6_SRC_4 | MLC6_SRC_3 | MLC6_SRC_2 | MLC6_SRC_1 | MLC6_SRC_0 |
| MLC7_SRC | 76h | MLC7_SRC_7 | MLC7_SRC_6 | MLC7_SRC_5 | MLC7_SRC_4 | MLC7_SRC_3 | MLC7_SRC_2 | MLC7_SRC_1 | MLC7_SRC_0 |
| MLC8_SRC | 77h | MLC8_SRC_7 | MLC8_SRC_6 | MLC8_SRC_5 | MLC8_SRC_4 | MLC8_SRC_3 | MLC8_SRC_2 | MLC8_SRC_1 | MLC8_SRC_0 |

## 2.2 Embedded advanced features pages

The table given below provides a list of the registers for the embedded advanced features page 0. These registers are accessible when PAGE_SEL[3:0] are set to 0000 in the PAGE_SEL register.

**Table 4. Embedded advanced features registers - page 0**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| FSM_EXT_SENSITIVITY_L | BAh | FSM_EXT_S_7 | FSM_EXT_S_6 | FSM_EXT_S_5 | FSM_EXT_S_4 | FSM_EXT_S_3 | FSM_EXT_S_2 | FSM_EXT_S_1 | FSM_EXT_S_0 |
| FSM_EXT_SENSITIVITY_H | BBh | FSM_EXT_S_15 | FSM_EXT_S_14 | FSM_EXT_S_13 | FSM_EXT_S_12 | FSM_EXT_S_11 | FSM_EXT_S_10 | FSM_EXT_S_9 | FSM_EXT_S_8 |
| FSM_EXT_OFFX_L | C0h | FSM_EXT_OFFX_7 | FSM_EXT_OFFX_6 | FSM_EXT_OFFX_5 | FSM_EXT_OFFX_4 | FSM_EXT_OFFX_3 | FSM_EXT_OFFX_2 | FSM_EXT_OFFX_1 | FSM_EXT_OFFX_0 |
| FSM_EXT_OFFX_H | C1h | FSM_EXT_OFFX_15 | FSM_EXT_OFFX_14 | FSM_EXT_OFFX_13 | FSM_EXT_OFFX_12 | FSM_EXT_OFFX_11 | FSM_EXT_OFFX_10 | FSM_EXT_OFFX_9 | FSM_EXT_OFFX_8 |
| FSM_EXT_OFFY_L | C2h | FSM_EXT_OFFY_7 | FSM_EXT_OFFY_6 | FSM_EXT_OFFY_5 | FSM_EXT_OFFY_4 | FSM_EXT_OFFY_3 | FSM_EXT_OFFY_2 | FSM_EXT_OFFY_1 | FSM_EXT_OFFY_0 |
| FSM_EXT_OFFY_H | C3h | FSM_EXT_OFFY_15 | FSM_EXT_OFFY_14 | FSM_EXT_OFFY_13 | FSM_EXT_OFFY_12 | FSM_EXT_OFFY_11 | FSM_EXT_OFFY_10 | FSM_EXT_OFFY_9 | FSM_EXT_OFFY_8 |
| FSM_EXT_OFFZ_L | C4h | FSM_EXT_OFFZ_7 | FSM_EXT_OFFZ_6 | FSM_EXT_OFFZ_5 | FSM_EXT_OFFZ_4 | FSM_EXT_OFFZ_3 | FSM_EXT_OFFZ_2 | FSM_EXT_OFFZ_1 | FSM_EXT_OFFZ_0 |
| FSM_EXT_OFFZ_H | C5h | FSM_EXT_OFFZ_15 | FSM_EXT_OFFZ_14 | FSM_EXT_OFFZ_13 | FSM_EXT_OFFZ_12 | FSM_EXT_OFFZ_11 | FSM_EXT_OFFZ_10 | FSM_EXT_OFFZ_9 | FSM_EXT_OFFZ_8 |
| FSM_EXT_MATRIX_XX_L | C6h | FSM_EXT_MAT_XX_7 | FSM_EXT_MAT_XX_6 | FSM_EXT_MAT_XX_5 | FSM_EXT_MAT_XX_4 | FSM_EXT_MAT_XX_3 | FSM_EXT_MAT_XX_2 | FSM_EXT_MAT_XX_1 | FSM_EXT_MAT_XX_0 |
| FSM_EXT_MATRIX_XX_H | C7h | FSM_EXT_MAT_XX_15 | FSM_EXT_MAT_XX_14 | FSM_EXT_MAT_XX_13 | FSM_EXT_MAT_XX_12 | FSM_EXT_MAT_XX_11 | FSM_EXT_MAT_XX_10 | FSM_EXT_MAT_XX_9 | FSM_EXT_MAT_XX_8 |
| FSM_EXT_MATRIX_XY_L | C8h | FSM_EXT_MAT_XY_7 | FSM_EXT_MAT_XY_6 | FSM_EXT_MAT_XY_5 | FSM_EXT_MAT_XY_4 | FSM_EXT_MAT_XY_3 | FSM_EXT_MAT_XY_2 | FSM_EXT_MAT_XY_1 | FSM_EXT_MATXY_0 |
| FSM_EXT_MATRIX_XY_H | C9h | FSM_EXT_MAT_XY_15 | FSM_EXT_MAT_XY_14 | FSM_EXT_MAT_XY_13 | FSM_EXT_MAT_XY_12 | FSM_EXT_MAT_XY_11 | FSM_EXT_MAT_XY_10 | FSM_EXT_MAT_XY_9 | FSM_EXT_MAT_XY_8 |
| FSM_EXT_MATRIX_XZ_L | CAh | FSM_EXT_MAT_XZ_7 | FSM_EXT_MAT_XZ_6 | FSM_EXT_MAT_XZ_5 | FSM_EXT_MAT_XZ_4 | FSM_EXT_MAT_XZ_3 | MAG_SI_XZ_2 | FSM_EXT_MAT_XZ_1 | FSM_EXT_MAT_XZ_0 |
| FSM_EXT_MATRIX_XZ_H | CBh | FSM_EXT_MAT_XZ_15 | FSM_EXT_MAT_XZ_14 | FSM_EXT_MAT_XZ_13 | FSM_EXT_MAT_XZ_12 | FSM_EXT_MAT_XZ_11 | FSM_EXT_MAT_XZ_10 | FSM_EXT_MAT_XZ_9 | FSM_EXT_MAT_XZ_8 |
| FSM_EXT_MATRIX_YY_L | CCh | FSM_EXT_MAT_YY_7 | FSM_EXT_MAT_YY_6 | FSM_EXT_MAT_YY_5 | FSM_EXT_MAT_YY_4 | FSM_EXT_MAT_YY_3 | FSM_EXT_MAT_YY_2 | FSM_EXT_MAT_YY_1 | FSM_EXT_MAT_YY_0 |
| FSM_EXT_MATRIX_YY_H | CDh | FSM_EXT_MAT_YY_15 | FSM_EXT_MAT_YY_14 | FSM_EXT_MAT_YY_13 | FSM_EXT_MAT_YY_12 | FSM_EXT_MAT_YY_11 | FSM_EXT_MAT_YY_10 | FSM_EXT_MAT_YY_9 | FSM_EXT_MAT_YY_8 |
| FSM_EXT_MATRIX_YZ_L | CEh | FSM_EXT_MAT_YZ_7 | FSM_EXT_MAT_YZ_6 | FSM_EXT_MAT_YZ_5 | FSM_EXT_MAT_YZ_4 | FSM_EXT_MAT_YZ_3 | FSM_EXT_MAT_YZ_2 | FSM_EXT_MAT_YZ_1 | FSM_EXT_MAT_YZ_0 |
| FSM_EXT_MATRIX_YZ_H | CFh | FSM_EXT_MAT_YZ_15 | FSM_EXT_MAT_YZ_14 | FSM_EXT_MAT_YZ_13 | FSM_EXT_MAT_YZ_12 | FSM_EXT_MAT_YZ_11 | FSM_EXT_MAT_YZ_10 | FSM_EXT_MAT_YZ_9 | FSM_EXT_MAT_YZ_8 |
| FSM_EXT_MATRIX_ZZ_L | D0h | FSM_EXT_MAT_ZZ_7 | FSM_EXT_MAT_ZZ_6 | FSM_EXT_MAT_ZZ_5 | FSM_EXT_MAT_ZZ_4 | FSM_EXT_MAT_ZZ_3 | FSM_EXT_MAT_ZZ_2 | FSM_EXT_MAT_ZZ_1 | FSM_EXT_MAT_ZZ_0 |
| FSM_EXT_MATRIX_ZZ_H | D1h | FSM_EXT_MAT_ZZ_15 | FSM_EXT_MAT_ZZ_14 | FSM_EXT_MAT_ZZ_13 | FSM_EXT_MAT_ZZ_12 | FSM_EXT_MAT_ZZ_11 | FSM_EXT_MAT_ZZ_10 | FSM_EXT_MAT_ZZ_9 | FSM_EXT_MAT_ZZ_8 |
| EXT_CFG_A | D4h | 0 | EXT_Y_AXIS2 | EXT_Y_AXIS1 | EXT_Y_AXIS0 | 0 | EXT_Z_AXIS2 | EXT_Z_AXIS1 | EXT_Z_AXIS0 |
| EXT_CFG_B | D5h | 0 | 0 | 0 | 0 | 0 | EXT_X_AXIS2 | EXT_X_AXIS1 | EXT_X_AXIS0 |

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

**Table 5. Embedded advanced features registers - page 1**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| XL_HG_SENSITIVITY_L | 58h | XL_HG_S_7 | XL_HG_S_6 | XL_HG_S_5 | XL_HG_S_4 | XL_HG_S_3 | XL_HG_S_2 | XL_HG_S_1 | XL_HG_S_0 |
| XL_HG_SENSITIVITY_H | 59h | XL_HG_S_15 | XL_HG_S_14 | XL_HG_S_13 | XL_HG_S_12 | XL_HG_S_11 | XL_HG_S_10 | XL_HG_S_9 | XL_HG_S_8 |
| FSM_LC_TIMEOUT_L | 7Ah | FSM_LC_ TIMEOUT7 | FSM_LC_ TIMEOUT6 | FSM_LC_ TIMEOUT5 | FSM_LC_ TIMEOUT4 | FSM_LC_ TIMEOUT3 | FSM_LC_ TIMEOUT2 | FSM_LC_ TIMEOUT1 | FSM_LC_ TIMEOUT0 |
| FSM_LC_TIMEOUT_H | 7Bh | FSM_LC_ TIMEOUT15 | FSM_LC_ TIMEOUT14 | FSM_LC_ TIMEOUT13 | FSM_LC_ TIMEOUT12 | FSM_LC_ TIMEOUT11 | FSM_LC_ TIMEOUT10 | FSM_LC_ TIMEOUT9 | FSM_LC_ TIMEOUT8 |
| FSM_PROGRAMS | 7Ch | FSM_N_PROG7 | FSM_N_PROG6 | FSM_N_PROG5 | FSM_N_PROG4 | FSM_N_PROG3 | FSM_N_PROG2 | FSM_N_PROG1 | FSM_N_PROG0 |
| FSM_START_ADD_L | 7Eh | FSM_START7 | FSM_START6 | FSM_START5 | FSM_START4 | FSM_START3 | FSM_START2 | FSM_START1 | FSM_START0 |
| FSM_START_ADD_H | 7Fh | FSM_START15 | FSM_START714 | FSM_START13 | FSM_START12 | FSM_START11 | FSM_START10 | FSM_START9 | FSM_START8 |
| PEDO_CMD_REG | 83h | 0 | 0 | 0 | 0 | CARRY_ COUNT_EN | FP_ REJECTION_EN | 0 | 0 |
| PEDO_DEB_STEPS_CONF | 84h | DEB_STEP7 | DEB_STEP6 | DEB_STEP5 | DEB_STEP4 | DEB_STEP3 | DEB_STEP2 | DEB_STEP1 | DEB_STEP0 |
| PEDO_SC_DELTAT_L | D0h | 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 | D1h | PD_SC_15 | PD_SC_14 | PD_SC_13 | PD_SC_12 | PD_SC_11 | PD_SC_10 | PD_SC_9 | PD_SC_8 |
| MLC_EXT_SENSITIVITY_L | E8h | MLC_ EXT_S_7 | MLC_ EXT_S_6 | MLC_ EXT_S_5 | MLC_ EXT_S_4 | MLC_ EXT_S_3 | MLC_ EXT_S_2 | MLC_ EXT_S_1 | MLC_ EXT_S_0 |
| MLC_EXT_SENSITIVITY_H | E9h | MLC_ EXT_S_15 | MLC_ EXT_S_14 | MLC_ EXT_S_13 | MLC_ EXT_S_12 | MLC_ EXT_S_11 | MLC_ EXT_S_10 | MLC_ EXT_S_9 | MLC_ EXT_S_8 |

The following table provides a list of the registers for the embedded advanced features page 2. These registers are accessible when PAGE_SEL[3:0] are set to 0010 in the PAGE_SEL register.

**Table 6. Embedded advanced features registers - page 2**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| EXT_FORMAT | 00h | 0 | 0 | 0 | 0 | 0 | 0 | EXT_FORMAT_SEL | 0 |
| EXT_3BYTE_SENSITIVITY_L | 02h | EXT_3BYTE_S_7 | EXT_3BYTE_S_6 | EXT_3BYTE_S_5 | EXT_3BYTE_S_4 | EXT_3BYTE_S_3 | EXT_3BYTE_S_2 | EXT_3BYTE_S_1 | EXT_3BYTE_S_0 |
| EXT_3BYTE_SENSITIVITY_H | 03h | EXT_3BYTE_S_15 | EXT_3BYTE_S_14 | EXT_3BYTE_S_13 | EXT_3BYTE_S_12 | EXT_3BYTE_S_11 | EXT_3BYTE_S_10 | EXT_3BYTE_S_9 | EXT_3BYTE_S_8 |
| EXT_3BYTE_OFFSET_XL | 06h | EXT_3BYTE_OFF_7 | EXT_3BYTE_OFF_6 | EXT_3BYTE_OFF_5 | EXT_3BYTE_OFF_4 | EXT_3BYTE_OFF_3 | EXT_3BYTE_OFF_2 | EXT_3BYTE_OFF_1 | EXT_3BYTE_OFF_0 |
| EXT_3BYTE_OFFSET_L | 07h | EXT_3BYTE_OFF_15 | EXT_3BYTE_OFF_14 | EXT_3BYTE_OFF_13 | EXT_3BYTE_OFF_12 | EXT_3BYTE_OFF_11 | EXT_3BYTE_OFF_10 | EXT_3BYTE_OFF_9 | EXT_3BYTE_OFF_8 |
| EXT_3BYTE_OFFSET_H | 08h | EXT_3BYTE_OFF_23 | EXT_3BYTE_OFF_22 | EXT_3BYTE_OFF_21 | EXT_3BYTE_OFF_20 | EXT_3BYTE_OFF_19 | EXT_3BYTE_OFF_18 | EXT_3BYTE_OFF_17 | EXT_3BYTE_OFF_16 |

## 2.3 Sensor hub registers

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

**Table 7. Sensor hub registers**

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| SENSOR_HUB_1 | 02h | SensorHub1_7 | SensorHub1_6 | SensorHub1_5 | SensorHub1_4 | SensorHub1_3 | SensorHub1_2 | SensorHub1_1 | SensorHub1_0 |
| SENSOR_HUB_2 | 03h | SensorHub2_7 | SensorHub2_6 | SensorHub2_5 | SensorHub2_4 | SensorHub2_3 | SensorHub2_2 | SensorHub2_1 | SensorHub2_0 |
| SENSOR_HUB_3 | 04h | SensorHub3_7 | SensorHub3_6 | SensorHub3_5 | SensorHub3_4 | SensorHub3_3 | SensorHub3_2 | SensorHub3_1 | SensorHub3_0 |
| SENSOR_HUB_4 | 05h | SensorHub4_7 | SensorHub4_6 | SensorHub4_5 | SensorHub4_4 | SensorHub4_3 | SensorHub4_2 | SensorHub4_1 | SensorHub4_0 |
| SENSOR_HUB_5 | 06h | SensorHub5_7 | SensorHub5_6 | SensorHub5_5 | SensorHub5_4 | SensorHub5_3 | SensorHub5_2 | SensorHub5_1 | SensorHub5_0 |
| SENSOR_HUB_6 | 07h | SensorHub6_7 | SensorHub6_6 | SensorHub6_5 | SensorHub6_4 | SensorHub6_3 | SensorHub6_2 | SensorHub6_1 | SensorHub6_0 |
| SENSOR_HUB_7 | 08h | SensorHub7_7 | SensorHub7_6 | SensorHub7_5 | SensorHub7_4 | SensorHub7_3 | SensorHub7_2 | SensorHub7_1 | SensorHub7_0 |
| SENSOR_HUB_8 | 09h | SensorHub8_7 | SensorHub8_6 | SensorHub8_5 | SensorHub8_4 | SensorHub8_3 | SensorHub8_2 | SensorHub8_1 | SensorHub8_0 |
| SENSOR_HUB_9 | 0Ah | SensorHub9_7 | SensorHub9_6 | SensorHub9_5 | SensorHub9_4 | SensorHub9_3 | SensorHub9_2 | SensorHub9_1 | SensorHub9_0 |
| SENSOR_HUB_10 | 0Bh | SensorHub10_7 | SensorHub10_6 | SensorHub10_5 | SensorHub10_4 | SensorHub10_3 | SensorHub10_2 | SensorHub10_1 | SensorHub10_0 |
| SENSOR_HUB_11 | 0Ch | SensorHub11_7 | SensorHub11_6 | SensorHub11_5 | SensorHub11_4 | SensorHub11_3 | SensorHub11_2 | SensorHub11_1 | SensorHub11_0 |
| SENSOR_HUB_12 | 0Dh | SensorHub12_7 | SensorHub12_6 | SensorHub12_5 | SensorHub12_4 | SensorHub12_3 | SensorHub12_2 | SensorHub12_1 | SensorHub12_0 |
| SENSOR_HUB_13 | 0Eh | SensorHub13_7 | SensorHub13_6 | SensorHub13_5 | SensorHub13_4 | SensorHub13_3 | SensorHub13_2 | SensorHub13_1 | SensorHub13_0 |
| SENSOR_HUB_14 | 0Fh | SensorHub14_7 | SensorHub14_6 | SensorHub14_5 | SensorHub14_4 | SensorHub14_3 | SensorHub14_2 | SensorHub14_1 | SensorHub14_0 |
| SENSOR_HUB_15 | 10h | SensorHub15_7 | SensorHub15_6 | SensorHub15_5 | SensorHub15_4 | SensorHub15_3 | SensorHub15_2 | SensorHub15_1 | SensorHub15_0 |
| SENSOR_HUB_16 | 11h | SensorHub16_7 | SensorHub16_6 | SensorHub16_5 | SensorHub16_4 | SensorHub16_3 | SensorHub16_2 | SensorHub16_1 | SensorHub16_0 |
| SENSOR_HUB_17 | 12h | SensorHub17_7 | SensorHub17_6 | SensorHub17_5 | SensorHub17_4 | SensorHub17_3 | SensorHub17_2 | SensorHub17_1 | SensorHub17_0 |
| SENSOR_HUB_18 | 13h | SensorHub18_7 | SensorHub18_6 | SensorHub18_5 | SensorHub18_4 | SensorHub18_3 | SensorHub18_2 | SensorHub18_1 | SensorHub18_0 |
| CONTROLLER_CONFIG | 14h | RST_CONTROLLER _REGS | WRITE_ONCE | START_CONFIG | PASS_ THROUGH_MODE | 0 | CONTROLLER_ON | AUX_SENS_ON1 | AUX_SENS_ON0 |
| TGT0_ADD | 15h | target0_add6 | target0_add5 | target0_add4 | target0_add3 | target0_add2 | target0_add1 | target0_add0 | rw_0 |
| TGT0_SUBADD | 16h | target0_reg7 | target0_reg6 | target0_reg5 | target0_reg4 | target0_reg3 | target0_reg2 | target0_reg1 | target0_reg0 |
| TGT0_CONFIG | 17h | SHUB_ODR_2 | SHUB_ODR_1 | SHUB_ODR_0 | 0 | BATCH_EXT _SENS_0_EN | target0_numop2 | target0_numop1 | target0_numop0 |
| TGT1_ADD | 18h | target1_add6 | target1_add5 | target1_add4 | target1_add3 | target1_add2 | target1_add1 | target1_add0 | r_1 |
| TGT1_SUBADD | 19h | target1_reg7 | target1_reg6 | target1_reg5 | target1_reg4 | target1_reg3 | target1_reg2 | target1_reg1 | target1_reg0 |
| TGT1_CONFIG | 1Ah | 0 | 0 | 0 | 1 | BATCH_EXT _SENS_1_EN | target1_numop2 | target1_numop1 | target1_numop0 |
| TGT2_ADD | 1Bh | target2_add6 | target2_add5 | target2_add4 | target2_add3 | target2_add2 | target2_add1 | target2_add0 | r_2 |
| TGT2_SUBADD | 1Ch | target2_reg7 | target2_reg6 | target2_reg5 | target2_reg4 | target2_reg3 | target2_reg2 | target2_reg1 | target2_reg0 |
| TGT2_CONFIG | 1Dh | 0 | 0 | 0 | 0 | BATCH_EXT _SENS_2_EN | target2_numop2 | target2_numop1 | target2_numop0 |

| Register name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| TGT3_ADD | 1Eh | target3_add6 | target3_add5 | target3_add4 | target3_add3 | target3_add2 | target3_add1 | target3_add0 | r_3 |
| TGT3_SUBADD | 1Fh | target3_reg7 | target3_reg6 | target3_reg5 | target3_reg4 | target3_reg3 | target3_reg2 | target3_reg1 | target3_reg0 |
| TGT3_CONFIG | 20h | 0 | 0 | 0 | 0 | BATCH_EXT_SENS_3_EN | target3_numop2 | target3_numop1 | target3_numop0 |
| DATAWRITE_TGT0 | 21h | target0_dataw7 | target0_dataw6 | target0_dataw5 | target0_dataw4 | target0_dataw3 | target0_dataw2 | target0_dataw1 | target0_dataw0 |
| STATUS_CONTROLLER | 22h | WR_ONCE_DONE | TARGET3_NACK | TARGET2_NACK | TARGET1_NACK | TARGET0_NACK | 0 | 0 | SENS_HUB_ENDOP |

# 3 Operating modes

The device offers a wide VDD voltage range from 1.71 V to 3.6 V and a VDDIO range from 1.08 V to 3.6 V. The power-on sequence is not restricted. The VDD/VDDIO pins can be set to either the power supply level or to ground level (they must not be left floating) and no specific sequence is required for powering them on.

In order to avoid potential conflicts, during the power-on sequence it is recommended to set the lines (on the host side) connected to the device I/O pins floating or connected to ground, until VDDIO is set. After VDDIO is set, the lines connected to the I/O pins have to be configured according to their default status described in Table 1. 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.

When the VDD power supply is applied, the device performs a 10 ms (maximum) boot procedure to load the trimming parameters. After the boot is completed, the low-*g* accelerometer, the high-*g* accelerometer, and the gyroscope are all automatically configured in power-down mode. To guarantee proper power-off of the device, it is recommended to maintain the duration of the VDD line to GND for at least 100 µs. When all the sensors are configured in power-down mode, the digital interfaces (I²C, MIPI I3C®, and SPI) are still active to allow communication with the device.

The low-*g* accelerometer and the gyroscope can be configured independently. When both the low-*g* accelerometer and gyroscope are on, the low-*g* accelerometer is synchronized with the gyroscope, and the data rates of the two sensors are integer multiples of each other.

The low-*g* accelerometer can be configured in one of the following power modes:

- Power-down mode
- Low-power mode (three different modes are available, depending on the number of averaged measurements)
- Normal mode
- High-performance mode
- High-accuracy ODR mode
- ODR-triggered mode

The gyroscope can be configured in one of the following power modes:

- Power-down mode
- Sleep mode
- Low-power mode
- High-performance mode
- High-accuracy ODR mode
- ODR-triggered mode

When the high-*g* accelerometer is used, the low-*g* accelerometer must be configured in high-performance mode or in high-accuracy ODR mode. It is possible to enable the high-*g* accelerometer in standalone mode (high-*g* accelerometer only). The ODRs are independent between the low-*g* and high-*g* accelerometers. There are no limitations on the selectable operating mode of the gyroscope when the high-*g* channel is enabled.

*Note: If the low-g accelerometer is configured in high-performance mode, all the available ODRs are supported both for the low-g and the high-g accelerometer (including ODR = 0 - power-down mode).*

*Note: If the low-g accelerometer is intended to be turned on while the high-g accelerometer is intended to be turned off, it is mandatory to turn the low-g on and wait at least 300 µs before turning the high-g accelerometer off. Only in case the low-g accelerometer is intended to be turned on in any low-power or normal modes, is it mandatory to turn the high-g accelerometer off and wait at least 300 µs before turning the low-g accelerometer on, since the low-g accelerometer low-power or normal modes are not compatible with the high-g accelerometer.*

*Note: If the high-g accelerometer is intended to be turned on while the low-g accelerometer is intended to be turned off, it is mandatory to turn the high-g on and wait at least 300 µs before turning the low-g accelerometer off. Only in case the low-g accelerometer is on in any low-power or normal modes, is it mandatory to turn the low-g accelerometer off and wait at least 4.5 ms before turning the high-g accelerometer on, since the low-g accelerometer low-power or normal modes are not compatible with the high-g accelerometer.*

The high-*g* accelerometer can be configured in one of the following power modes:

- Power-down mode
- High-performance mode
- High-accuracy ODR mode

## 3.1 Low-*g* accelerometer power modes and output data rates

The power mode and the output data rate of the low-*g* accelerometer can be selected using the CTRL1 register.

When the low-*g* accelerometer is configured in **power-down** mode, almost all internal blocks of the low-*g* accelerometer are switched off to minimize power consumption. The content of the configuration registers is preserved and the output data registers are not updated, keeping the last data sampled in memory before going into power-down mode.

When the low-*g* accelerometer is configured in **low-power** mode, its reading chain is automatically turned on and off to optimize the supply current. Three different low-power modes are available, based on the number of measurements that are averaged for the sample generation:

- Low-power mode 1 (LPM1), where two measurements are averaged
- Low-power mode 2 (LPM2), where four measurements are averaged
- Low-power mode 3 (LPM3), where eight measurements are averaged

Increasing the number of averaged measurements allows reducing the noise, while decreasing them allows reducing the supply current.

In the low-power modes, the antialiasing filter is disabled and the low-*g* accelerometer ODR is selectable up to 240 Hz.

*Note: If the low-g accelerometer is configured in any low-power mode, its ODR is intended to be changed and the gyroscope is intended to be turned on right after the ODR change, it is mandatory to wait at least 3.5 ms between changing the accelerometer ODR and turning the gyroscope on.*

When the low-*g* accelerometer is configured in **normal** mode, its reading chain is always on. The antialiasing filter is enabled and the low-*g* accelerometer ODR is selectable up to 1920 Hz. Normal mode provides a balanced trade-off between noise and supply current.

When the low-*g* accelerometer is configured in **high-performance** mode, its reading chain is always on. The antialiasing filter is enabled and the low-*g* accelerometer ODR is selectable up to 7680 Hz. High-performance mode provides the best performance in terms of noise.

When the low-*g* accelerometer is configured in **high-accuracy ODR** (HAODR) mode, its reading chain is always on. High-accuracy ODR mode provides the best performance in terms of noise (same as high-performance mode) and typically reduces the part-to-part ODR variation. HAODR mode is described in Section 3.4: HAODR mode.

When the low-*g* accelerometer is configured in **ODR-triggered** mode, its reading chain is always on. The antialiasing filter is enabled and the low-*g* accelerometer ODR can be fine-tuned by the user by means of an external reference signal. ODR-triggered mode provides the best performance in terms of noise (same as high-performance mode) and the additional capability to synchronize the low-*g* accelerometer ODR with the external reference signal. ODR-triggered mode is described in Section 3.5: ODR-triggered mode.

Table 8 summarizes the available power modes based on the OP_MODE_XL bits of the CTRL1 register. The power-down mode is selected if ODR_XL = 0000, regardless of the configuration of the OP_MODE_XL bits.

**Table 8. Low-*g* accelerometer power modes**

| OP_MODE_XL_[2:0] | Power mode |
|---|---|
| 000 | High-performance (default) |
| 001 | High-accuracy ODR |
| 011 | ODR-triggered |
| 100 | Low-power mode 1 |
| 101 | Low-power mode 2 |
| 110 | Low-power mode 3 |
| 111 | Normal |

Table 9 summarizes the available ODR values based on the ODR_XL bits of the CTRL1 register.

**Table 9. Low-*g* accelerometer ODR**

| ODR_XL_[3:0] | ODR selection [Hz] | High-performance | High-accuracy ODR (HAODR_SEL_[1:0] = 00) | Normal | Low-power mode 1 / 2 / 3 |
|---|---|---|---|---|---|
| 0000 | Power-down (default) | ✓ | ✓ | ✓ | ✓ |
| 0001 | 1.875 | | | | ✓ |
| 0010 | 7.5 | ✓ | | ✓ | |
| 0011 | 15 | ✓ | ✓ | ✓ | ✓ |
| 0100 | 30 | ✓ | ✓ | ✓ | ✓ |
| 0101 | 60 | ✓ | ✓ | ✓ | ✓ |
| 0110 | 120 | ✓ | ✓ | ✓ | ✓ |
| 0111 | 240 | ✓ | ✓ | ✓ | ✓ |
| 1000 | 480 | ✓ | ✓ | ✓ | |
| 1001 | 960 | ✓ | ✓ | ✓ | |
| 1010 | 1920 | ✓ | ✓ | ✓ | |
| 1011 | 3840 | ✓ | ✓ | | |
| 1100 | 7680 | ✓ | ✓ | | |

## 3.2 High-*g* accelerometer power modes and output data rate

When the high-*g* accelerometer is enabled, the low-*g* accelerometer must be configured in high-performance or in high-accuracy ODR mode. The low-power, normal, and ODR-triggered modes of the low-*g* accelerometer are not compatible with the high-*g* accelerometer.

When the high-*g* accelerometer is configured in **power-down** mode, almost all internal blocks of the high-*g* accelerometer are switched off to minimize the power consumption. The content of the configuration registers is preserved and the output data registers are not updated, keeping the last data sampled in memory before going into power-down mode.

When the high-*g* accelerometer is configured in **high-performance** mode, its reading chain is always on. The antialiasing filter is enabled and the high-*g* accelerometer ODR can be configured using register CTRL1_XL_HG up to 7680 Hz. Its output can be read in registers from UI_OUTX_L_A_HG to UI_OUTZ_H_A_HG by setting the XL_HG_REGOUT_EN bit to 1 in the CTRL1_XL_HG register.

When the high-*g* accelerometer is configured in **high-accuracy ODR** (HAODR) mode, its reading chain is always on. High-accuracy ODR mode provides the best performance in terms of noise (same as high-performance mode) and typically reduces the part-to-part ODR variation. HAODR mode is described in Section 3.4: HAODR mode.

**Table 10. High-*g* accelerometer ODR**

| ODR_XL_HG [2:0] | ODR selection [Hz] |
|---|---|
| 000 | Power-down (default) |
| 001 | Reserved |
| 010 | Reserved |
| 011 | 480 |
| 100 | 960 |
| 101 | 1920 |
| 110 | 3840 |
| 111 | 7680 |

## 3.3 Gyroscope power modes and output data rates

The power mode and the output data rate of the gyroscope can be selected using the CTRL2 register.

When the gyroscope is configured in **power-down** mode, almost all internal blocks of the gyroscope are switched off to minimize power consumption. The content of the configuration registers is preserved and the output data registers are not updated, keeping the last data sampled in memory before going into power-down mode.

When the gyroscope is in **sleep** mode, the circuitry that drives the oscillation of the gyroscope mass is active, but the reading chain is turned off. Compared to power-down mode, the turn-on time from sleep mode to any active mode is drastically reduced.

When the gyroscope is configured in **low-power** mode, the driving circuitry is always on, but the reading chain is automatically turned on and off to optimize the supply current. The gyroscope ODR is selectable up to 240 Hz.

When the gyroscope is configured in **high-performance** mode, its reading chain is always on. The gyroscope ODR is selectable up to 7680 Hz. High-performance mode provides the best performance in terms of noise.

When the gyroscope is configured in **high-accuracy ODR** (HAODR) mode, its reading chain is always on. High-accuracy ODR mode provides the best performance in terms of noise (same as high-performance mode) and typically reduces the part-to-part ODR variation. HAODR mode is described in Section 3.4: HAODR mode.

When the gyroscope is configured in **ODR-triggered** mode, its reading chain is always on. The gyroscope ODR can be fine-tuned by the user by means of an external reference signal. ODR-triggered mode provides the best performance in terms of noise (same as high-performance mode) and the additional capability to synchronize the gyroscope ODR with the external reference signal. ODR-triggered mode is described in Section 3.5: ODR-triggered mode.

Table 11 summarizes the available power modes based on the OP_MODE_G bits of the CTRL2 register. The power-down mode is selected if ODR_G = 0000, regardless of the configuration of the OP_MODE_G bits.

**Table 11. Gyroscope power modes**

| OP_MODE_G_[2:0] | Power mode |
|---|---|
| 000 | High-performance |
| 001 | High-accuracy ODR |
| 011 | ODR-triggered mode |
| 100 | Sleep |
| 101 | Low-power |

Table 12 summarizes the available ODR based on the ODR_G bits of the CTRL2 register.

**Table 12. Gyroscope ODR**

| ODR_G_[3:0] | ODR selection [Hz] | High-performance | High-accuracy ODR (HAODR_SEL_[1:0] = 00) | Low-power mode |
|---|---|---|---|---|
| 0000 | Power-down (default) | ✓ | ✓ | ✓ |
| 0010 | 7.5 | ✓ | | ✓ |
| 0011 | 15 | ✓ | ✓ | ✓ |
| 0100 | 30 | ✓ | ✓ | ✓ |
| 0101 | 60 | ✓ | ✓ | ✓ |
| 0110 | 120 | ✓ | ✓ | ✓ |
| 0111 | 240 | ✓ | ✓ | ✓ |
| 1000 | 480 | ✓ | ✓ | |
| 1001 | 960 | ✓ | ✓ | |
| 1010 | 1920 | ✓ | ✓ | |
| 1011 | 3840 | ✓ | ✓ | |
| 1100 | 7680 | ✓ | ✓ | |

Table 13. **Gyroscope ODR selection in high-accuracy ODR mode**

| ODR_G_[3:0] | ODR [Hz] HAODR_SEL_[1:0] = 00 | ODR [Hz] HAODR_SEL_[1:0] = 01 | ODR [Hz] HAODR_SEL_[1:0] = 10 | ODR [Hz] HAODR_SEL_[1:0] = 11 |
|---|---|---|---|---|
| 0000 | Power-down | Power-down | Power-down | Power-down |
| 0001 | Reserved | Reserved | Reserved | Reserved |
| 0010 | Reserved | Reserved | Reserved | Reserved |
| 0011 | 15 | 15.625 | 12.5 | 13 |
| 0100 | 30 | 31.25 | 25 | 26 |
| 0101 | 60 | 62.5 | 50 | 52 |
| 0110 | 120 | 125 | 100 | 104 |
| 0111 | 240 | 250 | 200 | 208 |
| 1000 | 480 | 500 | 400 | 417 |
| 1001 | 960 | 1000 | 800 | 833 |
| 1010 | 1920 | 2000 | 1600 | 1667 |
| 1011 | 3840 | 4000 | 3200 | 3333 |
| 1100 | 7680 | 8000 | 6400 | 6667 |
| Others | Reserved | Reserved | Reserved | Reserved |

## 3.4 HAODR mode

Every device has its own ODR frequency (due to natural spreads). The LSM6DSV80X device provides a way to reduce the part-to-part ODR variation for the low-*g* accelerometer, high-*g* accelerometer, and gyroscope output data by enabling the high-accuracy ODR (HAODR) mode. When HAODR mode is enabled, the antialiasing filter is used and four sets of ODRs are available, based on the value of the HAODR_SEL_[1:0] bits in the HAODR_CFG register, as shown in Table 14 and in Table 15. The supply current in HAODR mode is slightly higher compared to high-performance mode, as shown in Table 17. Low-*g* accelerometer and gyroscope supply current (@VDD = 1.8 V, T = 25°C) and in Table 18. Low-*g* accelerometer, high-*g* accelerometer, and gyroscope supply current (@VDD = 1.8 V, T = 25°C).

If high-accuracy ODR mode is intended to be used, the following limitations must be considered:

- When HAODR mode is intended to be used for one sensor (low-*g* accelerometer or gyroscope), the other sensor has to be configured in HAODR mode too. The high-*g* is automatically set in HAODR mode when it is enabled for the other two sensors.
- All the sensors must be set in power-down mode (ODR_XL = 0000, ODR_G = 0000, and ODR_XL_HG = 000) before enabling or disabling HAODR mode.
- If all the sensors are in HAODR mode and a runtime change that configures all the sensors in power-down mode occurs (ODR_XL = 0000, ODR_G = 0000, and ODR_XL_HG = 000), it is mandatory to wait at least 500 µs in power-down mode before turning any sensor on.
- If the low-*g* accelerometer and gyroscope are batched in FIFO with any BDR different from the ODR, the timing of the batched data may have a maximum time shift of 1 / BDR_XL or 1 / BDR_GY on a single sample when the gyroscope is internally settled after turning on or when it is turned off.
- If the low-*g* accelerometer is on and the gyroscope is off, the data-ready frequency is correct after 70 ms settling period.
- HAODR mode is not compatible with the pedometer, relative tilt, DRDY mask, or activity/inactivity functionality (only motion/stationary can be used).

**Table 14. Low-*g* accelerometer and gyroscope ODR selection in high-accuracy ODR mode**

| ODR_XL_[3:0] / ODR_G_[3:0] | ODR [Hz] HAODR_SEL_[1:0] = 00 | ODR [Hz] HAODR_SEL_[1:0] = 01 | ODR [Hz] HAODR_SEL_[1:0] = 10 | ODR [Hz] HAODR_SEL_[1:0] = 11 |
|---|---|---|---|---|
| 0000 | Power-down | Power-down | Power-down | Power-down |
| 0001 | Reserved | Reserved | Reserved | Reserved |
| 0010 | Reserved | Reserved | Reserved | Reserved |
| 0011 | 15 | 15.625 | 12.5 | 13 |
| 0100 | 30 | 31.25 | 25 | 26 |
| 0101 | 60 | 62.5 | 50 | 52 |
| 0110 | 120 | 125 | 100 | 104 |
| 0111 | 240 | 250 | 200 | 208 |
| 1000 | 480 | 500 | 400 | 417 |
| 1001 | 960 | 1000 | 800 | 833 |
| 1010 | 1920 | 2000 | 1600 | 1667 |
| 1011 | 3840 | 4000 | 3200 | 3333 |
| 1100 | 7680 | 8000 | 6400 | 6667 |
| Others | Reserved | Reserved | Reserved | Reserved |

**Table 15. High-*g* accelerometer ODR in high-accuracy ODR mode**

| ODR_XL_HG[2:0] | ODR [Hz] HAODR_SEL[1:0] = 00 | ODR [Hz] HAODR_SEL[1:0] = 01 | ODR [Hz] HAODR_SEL[1:0] = 10 | ODR [Hz] HAODR_SEL_[1:0] = 11 |
|---|---|---|---|---|
| 000 | Power-down | Power-down | Power-down | Power-down |
| 001 | Reserved | Reserved | Reserved | Reserved |
| 010 | Reserved | Reserved | Reserved | Reserved |
| 011 | 480 | 500 | 400 | 417 |
| 100 | 960 | 1000 | 800 | 833 |
| 101 | 1920 | 2000 | 1600 | 1667 |
| 110 | 3840 | 4000 | 3200 | 3333 |
| 111 | 7680 | 8000 | 6400 | 6667 |

## 3.5 ODR-triggered mode

Every device has its own ODR frequency (due to natural spreads). The LSM6DSV80X device provides a way to synchronize its data generation with an external hardware reference signal provided over the INT2 pin. The device is able to automatically align the frequency and phase to the edges of the reference signal.

ODR-triggered mode supports low-*g* accelerometer only, gyroscope only, and combo (low-*g* accelerometer and gyroscope) modes. If either the low-*g* accelerometer or the gyroscope is intended to be used in ODR-triggered mode, they must be both configured in ODR-triggered mode or power-down mode. When both the low-*g* accelerometer and gyroscope are enabled, the user must configure the same ODR on both the low-*g* accelerometer and gyroscope. It is not possible to select different ODRs for the low-*g* accelerometer and gyroscope; if different ODR values are set, the ODR configured for the gyroscope data is also applied to the low-*g* accelerometer data.

The low-*g* accelerometer and gyroscope full-scale configurations are totally independent, and they can be set in any combination.

*Note: ODR-triggered mode has to be enabled / disabled when the device is in power-down mode.*

*Note: When ODR-triggered mode is enabled, the 0001, 0010, and 1100 configurations of the ODR_XL_[3:0] bits in register CTRL1 and the 0010 and 1100 configurations of the ODR_G_[3:0] bits in register CTRL2 cannot be used.*

*Note: ODR-triggered mode is not compatible with the pedometer, relative tilt, SFLP, DRDY mask, or activity/inactivity functionality (only motion/stationary can be used).*

When using the ODR-triggered mode, the lock is reached (and the filtering chains are settled) after 4 clock periods of the external reference signal.

When using additional digital filters (for example, the low-*g* accelerometer LPF2 or HP filter, or gyroscope LPF1 filter), their settling time must be considered in addition to the filtering chain settling time indicated above (see Section 3.8: Low-*g* accelerometer bandwidth and Section 3.12: Gyroscope bandwidth for more details about the low-*g* accelerometer and gyroscope bandwidth and turn-on/off time).

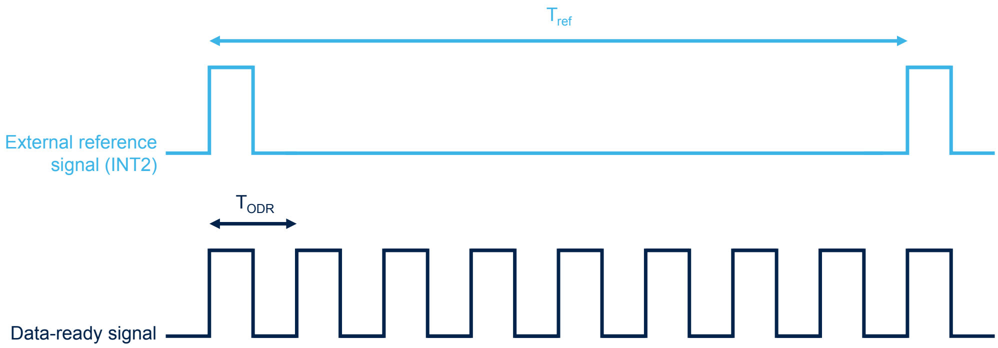
The selectable output date rates ($ODR_{sel}$) are different with respect to the regular ODRs set. The external reference signal must be provided with a period, which is an even multiple of the desired ODR period. The desired ODR period can have a maximum deviation of ±33% with respect to the selected $ODR_{sel}$. The desired number of samples in one external reference signal period must be configured through the ODR_TRIG_N_ODR_[7:0] bits of the ODR_TRIG_CFG register. The value of the ODR_TRIG_N_ODR_[7:0] bits can span from a minimum of 4 to a maximum of 255 and its resolution is 2 samples (therefore, from a minimum of 8 samples to a maximum of 510 samples). The $ODR_{sel}$ set available for ODR-triggered mode, the corresponding minimum number of samples for each ODR, and the corresponding minimum period for the external reference signal are indicated in Table 16.

**Table 16. ODR-triggered mode configurability**

| ODR_XL_[3:0] / ODR_G_[3:0] | $ODR_{sel}$ [Hz] | $\frac{T_{ref}}{T_{ODR}}$ minimum ratio [# samples] | Minimum $T_{ref}$ period [ms] |
|---|---|---|---|
| 0011 | 12.5 | 8 | 640 |
| 0100 | 25 | 8 | 320 |
| 0101 | 50 | 8 | 160 |
| 0110 | 100 | 8 | 80 |
| 0111 | 200 | 8 | 40 |
| 1000 | 400 | 16 | 40 |
| 1001 | 800 | 32 | 40 |
| 1010 | 1600 | 64 | 40 |
| 1011 | 3200 | 128 | 40 |

Figure 2 shows an example of the external reference signal with respect to the internal data-ready signal when configuring ODR_TRIG_N_ODR_[7:0] to 4 (8 samples).

**Figure 2.** **External reference signal example (INT2_IN_LH = 1)**




The external reference signal pulse duration must be at least 5 µs and its polarity can be selected through the INT2_IN_LH bit of the CTRL4 register. If the INT2_IN_LH bit is set to 0, the ODR-triggered mode is sensitive to the falling edge of the external reference signal, otherwise it is sensitive to the rising edge.

The following procedure must be used for configuring the device in ODR-triggered mode.

1. Set both the sensors in power-down mode.
2. Configure the number of samples for each $T_{ref}$ period by setting the ODR_TRIG_N_ODR_[7:0] bits.
3. Set both sensors in ODR-triggered mode.
4. Configure the reference signal polarity.
5. Start the external reference signal on INT2.
6. Set the low-*g* accelerometer and/or gyroscope $ODR_{sel}$.

The following example configures the sensors in ODR-triggered combo mode at 100 Hz (10 ms time period).

| | |
|---|---|
| 1. Write 00h to CTRL1 register | // Set low-*g* accelerometer in power-down mode |
| 2. Write 00h to CTRL2 register | // Set gyroscope in power-down mode |
| 3. Write 04h to ODR_TRIG_CFG register | // Configure 8 samples per each $T_{ref}$ period |
| 4. Write 30h to CTRL1 register | // Set low-*g* accelerometer in ODR-triggered mode |
| 5. Write 30h to CTRL2 register | // Set gyroscope in ODR-triggered mode |
| 6. Write 01h to CTRL4 register | // ODR-triggered mode sensitive to rising edge |
| 7. Start external reference signal with an 80 ms period | // $T_{ref}$ = 80 ms = 10 ms * 8 samples |
| 8. Write 36h to CTRL1 register | // Set low-*g* accelerometer $ODR_{sel}$ to 100 Hz |
| 9. Write 36h to CTRL2 register | // Set gyroscope $ODR_{sel}$ to 100 Hz |

100 Hz has a 0% deviation with respect to the selected ODR configured through the ODR_XL_[3:0] bits of the CTRL1 register and ODR_G_[3:0] bits of the CTRL2 register. The user could, using the same configuration listed above, achieve an ODR of 100 Hz ±33% by fine-tuning the external reference signal period ($T_{ref}$).

## 3.6 Supply current

Table 17 shows the typical values of supply current for the different operating modes.

**Table 17. Low-*g* accelerometer and gyroscope supply current (@VDD = 1.8 V, T = 25°C)**

| ODR [Hz] | Low-*g* accelerometer only [µA] | Gyroscope only [µA] | Combo low-*g* accelerometer + gyroscope [µA] |
|---|---|---|---|
| Power-down | 2.9 | 2.9 | 2.9 |
| 1.875 Hz (low-power mode 1) | 4.5 | - | - |
| 15 Hz (low-power mode 1) | 7.6 | - | - |
| 30 Hz (low-power mode 1) | 11.0 | - | - |
| 60 Hz (low-power mode 1) | 18.0 | - | - |
| 120 Hz (low-power mode 1) | 30.3 | - | - |
| 240 Hz (low-power mode 1) | 55.7 | - | - |
| 1.875 Hz (low-power mode 2) | 4.6 | - | - |
| 7.5 Hz (low-power) | - | 257 | |
| 15 Hz (low-power mode 2) | 8.3 | 263 | - |
| 30 Hz (low-power mode 2) | 12.3 | 273 | - |
| 60 Hz (low- power mode 2) | 21.0 | 293 | - |
| 120 Hz (low-power mode 2) | 36.3 | 334 | - |
| 240 Hz (low-power mode 2) | 67.2 | 415 | - |
| 1.875 Hz (low-power mode 3) | 4.9 | - | - |
| 15 Hz (low-power mode 3) | 9.8 | - | - |
| 30 Hz (low-power mode 3) | 15.5 | - | - |
| 60 Hz (low-power mode 3) | 26.5 | - | - |
| 120 Hz (low-power mode 3) | 47.6 | - | - |
| 240 Hz (low-power mode 3) | 90.2 | - | - |
| All ODRs (normal mode) | 115 | - | - |
| All ODRs (high-performance mode) | 200 | 550 | 670 |
| All ODRs (high-accuracy ODR mode) | 224 | 574 | 716 |

**Table 18. Low-*g* accelerometer, high-*g* accelerometer, and gyroscope supply current (@VDD = 1.8 V, T = 25°C)**

| ODR [Hz] | High-*g* accelerometer [µA] | Combo low-*g* accelerometer + high-*g* accelerometer [µA] | Combo low-*g* accelerometer + high-*g* accelerometer + gyroscope [µA] |
|---|---|---|---|
| All ODRs (high-performance mode) | 225 | 318 | 800 |
| All ODRs (high-accuracy ODR mode) | 245 | 358 | 848 |

*Note: Use standard supply current measurement equipment or the MEMS Studio sensor evaluation tool to measure the supply current of any configuration not indicated in the tables above.*

## 3.7 Connection modes

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

- **Mode 1**: it is the connection mode enabled by default. The I²C target interface, MIPI I3C® target interface, or SPI (3- / 4-wire) serial interface is available.
- **Mode 2**: it is the sensor hub mode. The I²C target interface, MIPI I3C® target interface, or SPI (3- / 4-wire) serial interface and I²C interface controller for external sensor connections are available. This connection mode is described in Section 7: Mode 2 - sensor hub mode.

## 3.8 Low-*g* accelerometer bandwidth

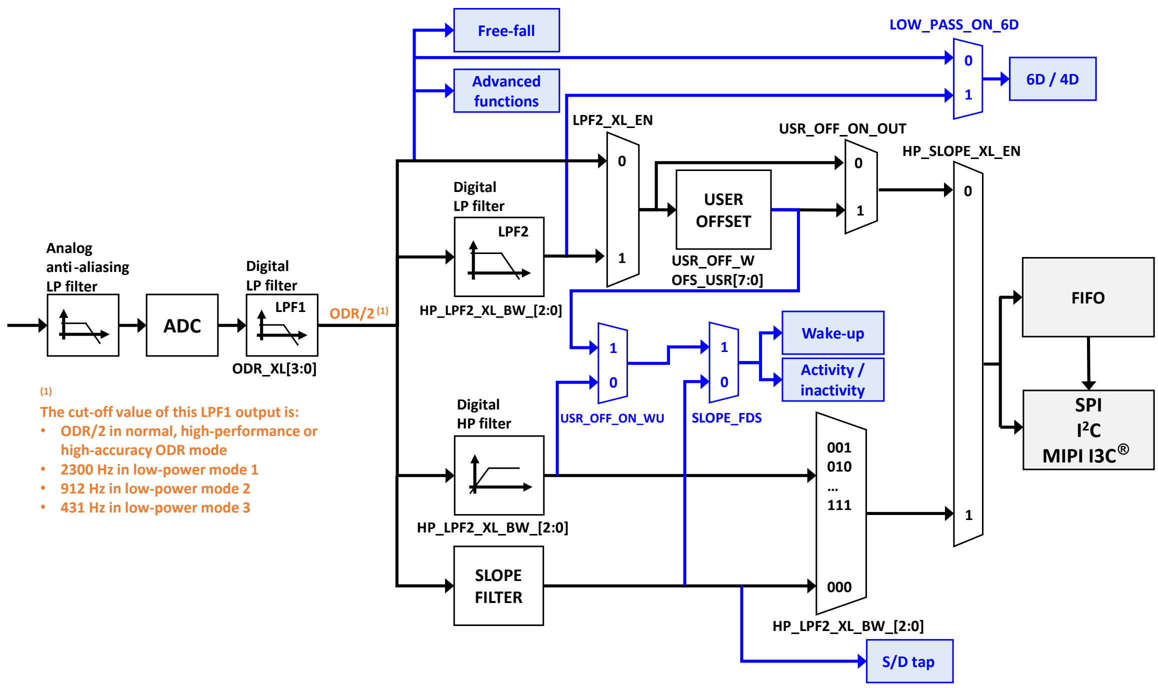
The low-*g* accelerometer sampling chain is represented by a cascade of four main blocks: an analog antialiasing low-pass filter, an ADC converter, a digital low-pass filter (LPF1), and the composite group of digital filters.

Figure 3. Low-*g* accelerometer filtering chain (UI path) shows the low-*g* accelerometer sampling chain on the UI path.

The analog signal coming from the mechanical parts is filtered by an analog antialiasing low-pass filter before being converted by the ADC. The antialiasing filter is not enabled in the low-power modes. The digital LPF1 filter provides different cutoff values based on the low-*g* accelerometer mode selected:

- ODR / 2 when the low-*g* accelerometer is configured in normal, high-performance, or high-accuracy ODR mode
- 2300 Hz when the low-*g* accelerometer is configured in low-power mode 1
- 912 Hz when the low-*g* accelerometer is configured in low-power mode 2
- 431 Hz when the low-*g* accelerometer is configured in low-power mode 3

Figure 3. **Low-*g* accelerometer filtering chain (UI path)**




The “Advanced functions” block in the figure above refers to the pedometer, step detector and step counter, significant motion, tilt, and SFLP functions, described in Section 6: Embedded functions, and also includes the finite state machine and the machine learning core.

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

The HP_LPF2_XL_BW_[2:0] bits in the CTRL8 register and the CTRL9 register can be used to configure the composite filter group and the overall bandwidth of the low-*g* accelerometer filtering chain, as shown in Table 19. Low-*g* accelerometer bandwidth selection in mode 1/2/3. Referring to this table, on the low-pass path side, the Bandwidth column refers to the LPF1 bandwidth if LPF2_XL_EN = 0; it refers to the LPF2 bandwidth if LPF2_XL_EN = 1. On the high-pass path side, the Bandwidth column refers to the slope filter bandwidth if HP_LPF2_XL_BW_[2:0] = 000; it refers to the HP filter bandwidth for all the other configurations.

Table 19. Low-*g* accelerometer bandwidth selection in mode 1/2/3 also provides the maximum (worst case) settling time in terms of samples to be discarded for the various configurations of the low-*g* accelerometer filtering chain.

**Table 19. Low-*g* accelerometer bandwidth selection in mode 1/2/3**

| HP_SLOPE_XL_EN | LPF2_XL_EN | HP_LPF2_XL_BW_[2:0] | Bandwidth | Max overall settling time(1) (samples to be discarded) |
|---|---|---|---|---|
| 0 (Low-pass path) | 0 | - | ODR / 2(2) | See Table 21 |
| | 1 | 000 | ODR / 4 | See Table 21 |
| | | 001 | ODR / 10 | 21 |
| | | 010 | ODR / 20 | 21 |
| | | 011 | ODR / 45 | 39 |
| | | 100 | ODR / 100 | 78 |
| | | 101 | ODR / 200 | 156 |
| | | 110 | ODR / 400 | 313 |
| | | 111 | ODR / 800 | 626 |
| 1 (High-pass path) | - | 000 | ODR / 4 (slope filter) | See Table 21 |
| | | 001 | ODR / 10 | 27 |
| | | 010 | ODR / 20 | 27 |
| | | 011 | ODR / 45 | 39 |
| | | 100 | ODR / 100 | 78 |
| | | 101 | ODR / 200 | 156 |
| | | 110 | ODR / 400 | 313 |
| | | 111 | ODR / 800 | 626 |

1. *Settling time @ 99% of the final value, taking into account all output data rates and all operating mode switches*
2. *This value is ODR / 2 when the low-g accelerometer is in high-performance mode, high-accuracy ODR mode and normal mode. It is equal to 2300 Hz when the low-g accelerometer is in low-power mode 1 (2 mean), 912 Hz in low-power mode 2 (4 mean) and 431 Hz in low-power mode 3 (8 mean).*

Setting the HP_SLOPE_XL_EN bit to 0, the low-pass path of the composite filter block is selected. If the LPF2_XL_EN bit is set to 0, no additional filter is applied. If the LPF2_XL_EN bit is set to 1, the LPF2 filter is applied in addition to LPF1, and the overall bandwidth of the low-*g* accelerometer chain can be set by configuring the HP_LPF2_XL_BW_[2:0] field of the CTRL8 register.

The LPF2 low-pass filter can also be used in the 6D/4D functionality by setting the LOW_PASS_ON_6D bit of the TAP_CFG0 register to 1.

Setting the HP_SLOPE_XL_EN bit to 1, the high-pass path of the composite filter block is selected. The HP_LPF2_XL_BW_[2:0] field is used in order to enable, in addition to the LPF1 filter, either the slope filter usage (when HP_LPF2_XL_BW_[2:0] = 000) or the digital high-pass filter (other HP_LPF2_XL_BW_[2:0] configurations). The HP_LPF2_XL_BW_[2:0] field is also used to select the cutoff frequencies of the HP filter.

The high-pass filter reference mode feature is available for the low-*g* accelerometer sensor: when this feature is enabled, the current X, Y, Z low-*g* accelerometer sample is internally stored and subtracted from all subsequent output values. In order to enable the reference mode, both the HP_REF_MODE_XL bit and the HP_SLOPE_XL_EN bit of the CTRL9 register have to be set to 1, and the value of the HP_LPF2_XL_BW_[2:0] field has to be different than 000. When the reference mode feature is enabled, both the LPF2 filter and the HP filter are not available. The first low-*g* accelerometer output data after enabling the reference mode has to be discarded.

The XL_FASTSETTL_MODE bit of the CTRL9 register enables the low-*g* accelerometer LPF2 or HPF fast-settling mode: the selected filter sets the first sample after writing this bit. This feature applies only upon device exit from power-down mode.

### 3.8.1 Low-*g* accelerometer slope filter

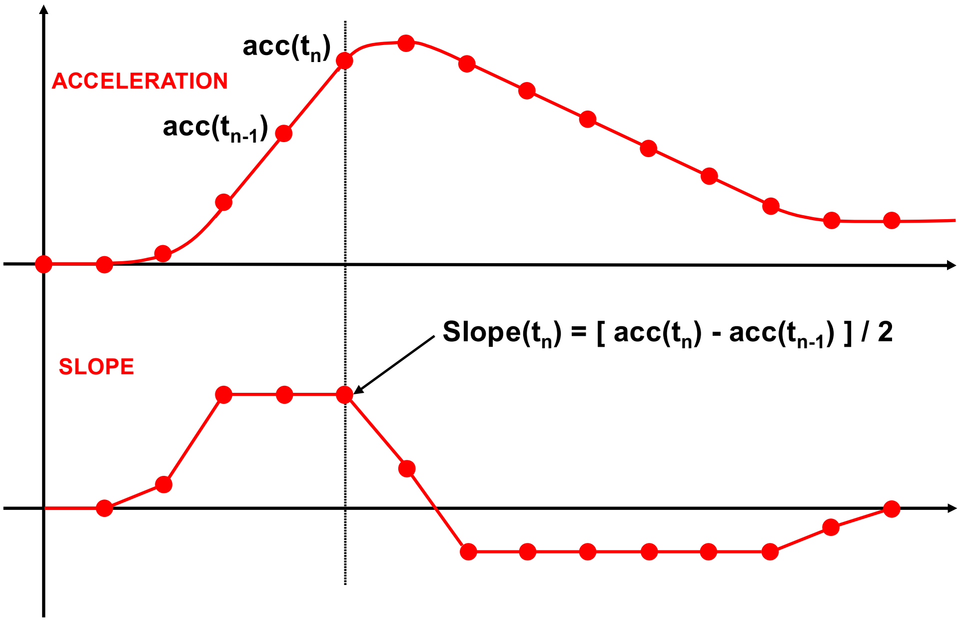
As shown in Figure 3. Low-*g* accelerometer filtering chain (UI path), the device embeds a digital slope filter, which can also be used for some embedded features such as single/double-tap recognition, wake-up detection and activity/inactivity.

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 4. Low-*g* accelerometer slope filter**




## 3.9 Low-*g* accelerometer turn-on/off time

The low-*g* accelerometer reading chain contains low-pass filtering to improve signal-to-noise performance and to reduce aliasing effects. For this reason, it is necessary to take into account the settling time of the filters when the low-*g* accelerometer power mode is switched or when the low-*g* accelerometer ODR is changed.

The low-*g* accelerometer chain settling time is dependent on the power mode and output data rate selected for the following configurations:

- LPF2 and HP filters disabled
- LPF2 or HP filter enabled with ODR / 4 bandwidth selection

For these two possible configurations, the maximum overall turn-on/off time to switch the low-*g* accelerometer power mode or low-*g* accelerometer ODR is the one shown below in Table 20 and Table 21.

*Note: The low-g accelerometer ODR timing is not impacted by power mode changes (the new configuration is active after the completion of the current period).*

**Table 20. Low-*g* accelerometer turn-on/off time (LPF2 and HP disabled)**

| Starting mode | Target mode | Max turn-on/off time [(1)] |
|---|---|---|
| Power-down | Low-power mode | See Table 21 |
| Power-down | Normal mode | See Table 21 |
| Power-down | High-performance mode | See Table 21 |
| Power-down | High-accuracy ODR mode | See Table 21 |
| Normal / low-power mode | High-performance mode | See Table 21 + discard 1 additional sample |
| High-performance / low-power mode | Normal mode | See Table 21 + discard 1 additional sample |
| High-performance / normal mode | Low-power mode | See Table 21 + discard 1 additional sample |
| Low-power mode | Low-power mode (ODR change) | See Table 21 |
| Normal mode | Normal mode (ODR change) | Discard 3 samples |
| High-performance mode | High-performance mode (ODR change) | Discard 4 samples |
| Low-power / normal / high-performance / high-accuracy ODR mode | Power-down | 1 µs |

1. *Settling time @ 99% of the final value*

**Table 21. Low-*g* accelerometer samples to be discarded**

| Target mode low-*g* accelerometer ODR [Hz] | Number of samples to be discarded (LPF2 and HP filters disabled) | | | | Number of samples to be discarded (LPF2 or HP filter enabled @ ODR / 4 bandwidth) | | | |
|---|---|---|---|---|---|---|---|---|
| | Low-power mode | Normal mode | High-performance mode | High-accuracy ODR mode | Low-power mode | Normal mode | High-performance mode | High-accuracy ODR mode |
| 1.875 Hz | 0 (first sample correct) | - | - | - | 1 | - | - | - |
| 7.5 Hz | - | 1 | 1 | - | - | 2 | 2 | - |
| 15 Hz | 0 (first sample correct) | 1 | 1 | 4 | 1 | 2 | 2 | 5 |
| 30 Hz | 0 (first sample correct) | 1 | 1 | 4 | 1 | 2 | 2 | 5 |
| 60 Hz | 0 (first sample correct) | 1 | 1 | 4 | 1 | 2 | 2 | 5 |
| 120 Hz | 0 (first sample correct) | 1 | 1 | 4 | 1 | 2 | 2 | 5 |
| 240 Hz | 0 (first sample correct) | 1 | 1 | 4 | 1 | 2 | 2 | 5 |
| 480 Hz | - | 1 | 1 | 4 | - | 2 | 2 | 5 |
| 960 Hz | - | 2 | 1 | 4 | - | 3 | 2 | 5 |
| 1920 Hz | - | 3 | 4 | 8 | - | 4 | 5 | 9 |
| 3840 Hz | - | - | 12 | 16 | - | - | 13 | 17 |
| 7680 Hz | - | - | 27 | 32 | - | - | 27 | 32 |

The overall settling time if LPF2 or the HP digital filters are enabled with a bandwidth different from ODR / 4 is indicated in Table 19. Low-*g* accelerometer bandwidth selection in mode 1/2/3.

## 3.10 High-*g* accelerometer bandwidth

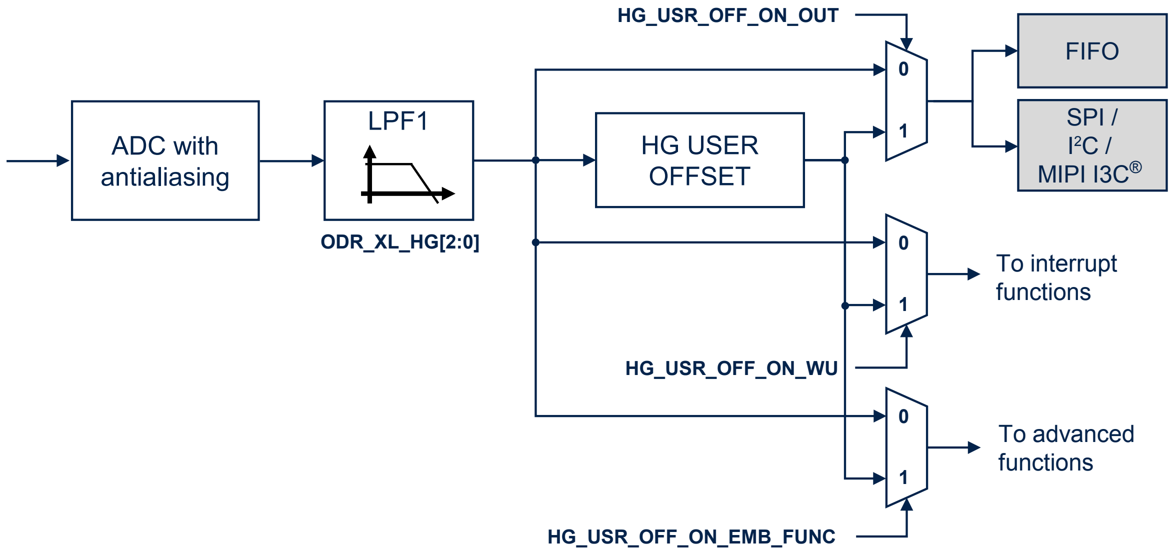
The high-*g* accelerometer sampling chain is represented by a cascade of three main blocks: an analog antialiasing low-pass filter, an ADC converter, and a digital low-pass filter (LPF1).

Figure 5. High-*g* accelerometer filtering chain shows the high-*g* 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. The digital LPF1 filter provides different cutoff values which determine the high-*g* accelerometer ODR. Refer to Section 3.2: High-*g* accelerometer power modes and output data rate for a complete description of the ODR configurations available.

A user-defined offset can be applied to the high-*g* on the UI/FIFO path by setting the HG_USR_OFF_ON_OUT bit to 1 in the CTRL1_XL_HG register. The same offset can be applied to the high-*g* interrupt functions (that is, high-*g* wake-up and shock detection) path by setting the HG_USR_OFF_ON_WU bit to 1 in the CTRL2_XL_HG register. The same offset can be applied to the high-*g* embedded functions (MLC, FSM) path by setting the HG_USR_OFF_ON_EMB_FUNC bit to 1 in the EMB_FUNC_CFG register. Refer to Section 4.7: High-*g* accelerometer offset registers for a detailed explanation about the user offset configuration, and refer to Section 5.8: High-*g* wake-up and shock detection for a detailed explanation of high-*g* wake-up and shock detection.

**Figure 5. High-*g* accelerometer filtering chain**




## 3.11 High-*g* accelerometer turn-on/off

The high-*g* accelerometer reading chain contains low-pass filtering to improve signal-to-noise performance and to reduce aliasing effects. For this reason, it is necessary to take into account the settling time of the filters when the high-*g* accelerometer power mode is turned on or when the high-*g* accelerometer ODR is changed.

Table 22 indicates the number of samples of the high-*g* accelerometer to be discarded depending on the selected ODR for both high-performance mode and high-accuracy ODR when exiting power-down mode.

*Note: Seven samples must be discarded to have a stable output for any ODR change in the selected mode.*

**Table 22. High-*g* accelerometer samples to be discarded**

| ODR [Hz] | High-performance mode | High-accuracy ODR mode |
|---|---|---|
| 480 Hz | 2 | 3 |
| 960 Hz | 2 | 3 |
| 1920 Hz | 3 | 4 |
| 3840 Hz | 4 | 5 |
| 7680 Hz | 8 | 8 |

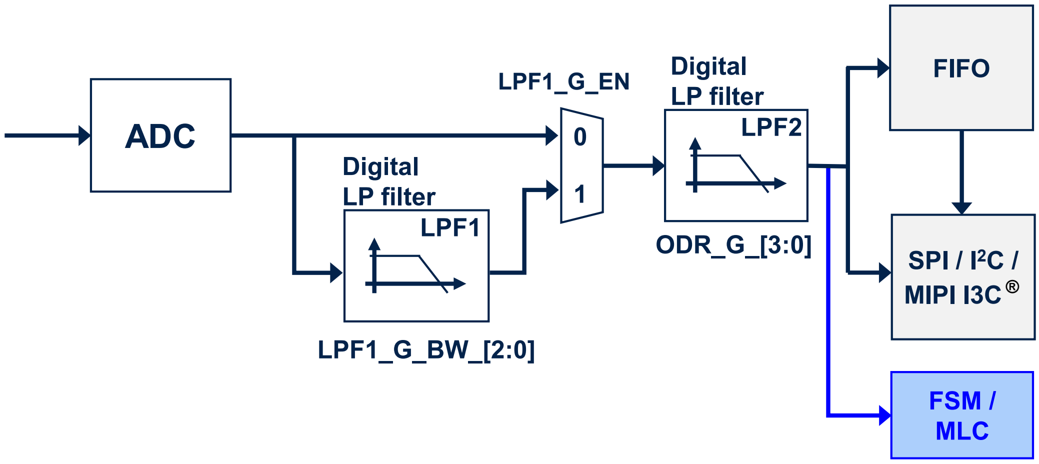
## 3.12 Gyroscope bandwidth

The gyroscope filtering chain depends on the connection mode in use.

When mode 1 or mode 2 is selected, the gyroscope filtering chain configuration is the one shown in Figure 6. It is a cascade of two filters: a selectable digital low-pass filter (LPF1) and a digital low-pass filter (LPF2).

The LPF1 filter is available in high-performance mode and in high-accuracy ODR mode. If the gyroscope is configured in low-power mode, the LPF1 filter is bypassed.

**Figure 6. Gyroscope digital chain - mode 1 (UI) and mode 2**




The digital LPF1 filter can be enabled by setting the LPF1_G_EN bit of the CTRL7 register to 1 and its bandwidth can be selected through the field LPF1_G_BW_[2:0] of the CTRL6 register.

The digital LPF2 filter cannot be configured by the user and its cutoff frequency depends on the selected gyroscope ODR. When the gyroscope ODR is equal to 7680 Hz, the LPF2 filter is bypassed.

The overall gyroscope bandwidth for different gyroscope ODR values and for different configurations of the LPF1_G_EN bit of the CTRL7 register and LPF1_G_BW_[2:0] of the CTRL6 register is summarized in the following table.

**Table 23. Gyroscope overall bandwidth selection in mode 1/2**

| Gyroscope ODR [Hz] | LPF1 | LPF2 | TOTAL LPF cutoff [Hz] (phase @ 20 Hz) |
|---|---|---|---|
| | LPF1_G_BW_[2:0] | | |
| 7.5 Hz | Bypassed | Enabled | 3.4 (-62.7° @ 2.5 Hz) |
| | 0xx | | 3.4 (-63.4° @ 2.5 Hz) |
| | 100 | | 3.4 (-64.6° @ 2.5 Hz) |
| | 101 | | 3.3 (-66.1° @ 2.5 Hz) |
| | 110 | | 3.3 (-67.7° @ 2.5 Hz) |
| | 111 | | 3.2 (-72.6° @ 2.5 Hz) |
| 15 Hz | Bypassed | Enabled | 6.6 (-65.3° @ 5 Hz) |
| | 0xx | | 6.6 (-66.7° @ 5 Hz) |
| | 100 | | 6.6 (-69.2° @ 5 Hz) |
| | 101 | | 6.6 (-72.1° @ 5 Hz) |
| | 110 | | 6.4 (-75.2° @ 5 Hz) |
| | 111 | | 5.9 (-84.6° @ 5 Hz) |
| 30 Hz | Bypassed | Enabled | 13.0 (-70.4° @ 10 Hz) |
| | 0xx | | 13.0 (-73.2° @ 10 Hz) |
| | 100 | | 13.0 (-78.3° @ 10 Hz) |

| Gyroscope ODR [Hz] | LPF1 LPF1_G_BW_[2:0] | LPF2 | TOTAL LPF cutoff [Hz] (phase @ 20 Hz) |
|---|---|---|---|
| 30 Hz | 101 | Enabled | 13.0 (-84.2° @ 10 Hz) |
| | 110 | | 11.6 (-89.7° @ 10 Hz) |
| | 111 | | 9.3 (-105° @ 10 Hz) |
| 60 Hz | Bypassed | Enabled | 24.6 (-80.6°) |
| | 0xx | | 24.6 (-86.1°) |
| | 100 | | 24.6 (-96.5°) |
| | 101 | | 24.6 (-109°) |
| | 110 | | 18.0 (-116°) |
| | 111 | | 12.1 (-135°) |
| 120 Hz | Bypassed | Enabled | 49.4 (-42.8°) |
| | 0xx | | 49.4 (-48.4°) |
| | 100 | | 49.4 (-58.7°) |
| | 101 | | 42.6 (-71.5°) |
| | 110 | | 24.2 (-77.9°) |
| | 111 | | 13.6 (-97.5°) |
| 240 Hz | Bypassed | Enabled | 96 (-24.8°) |
| | 0xx | | 96 (-30.4°) |
| | 100 | | 78.4 (-40.7°) |
| | 101 | | 53 (-53.6°) |
| | 110 | | 27.3 (-59.9°) |
| | 111 | | 14.2 (-79.5°) |
| 480 Hz | Bypassed | Enabled | 187 (-15.6°) |
| | 000 | | 175 (-21.1°) |
| | 001 | | 157 (-23.0°) |
| | 010 | | 131 (-25.9°) |
| | 011 | | 188 (-19.1°) |
| | 100 | | 94 (-31.5°) |
| | 101 | | 56.7 (-44.3°) |
| | 110 | | 28.4 (-50.7°) |
| | 111 | | 14.3 (-70.3°) |
| 960 Hz | Bypassed | Enabled | 342 (-10.9°) |
| | 000 | | 241 (-16.4°) |
| | 001 | | 195 (-18.3°) |
| | 010 | | 149 (-21.1°) |
| | 011 | | 310 (-14.4°) |
| | 100 | | 100 (-26.8°) |
| | 101 | | 57.9 (-39.6°) |
| | 110 | | 28.7 (-46.0°) |
| | 111 | | 14.4 (-65.6°) |
| 1920 Hz | Bypassed | Enabled | 491 (-8.3°) |
| | 000 | | 273 (-13.9°) |
| | 001 | | 210 (-15.8°) |
| | 010 | | 155 (-18.6°) |

| Gyroscope ODR [Hz] | LPF1 | LPF2 | TOTAL LPF cutoff [Hz] (phase @ 20 Hz) |
|---|---|---|---|
| | LPF1_G_BW_[2:0] | | |
| 1920 Hz | 011 | Enabled | 387 (-11.8°) |
| | 100 | | 101 (-24.3°) |
| | 101 | | 58.2 (-37.1°) |
| | 110 | | 28.8 (-43.5°) |
| | 111 | | 14.4 (-63.1°) |
| 3840 Hz | Bypassed | Enabled | 528 (-7.4°) |
| | 000 | | 280 (-13.0°) |
| | 001 | | 213 (-14.9°) |
| | 010 | | 156 (-17.7°) |
| | 011 | | 403 (-10.9°) |
| | 100 | | 102 (-23.3°) |
| | 101 | | 58.3 (-36.1°) |
| | 110[1] | | 28.8 (-42.5°) |
| | 111[1] | | 14.4 (-62.1°) |
| 7680 Hz | Bypassed | Disabled | 537 (-6.9°) |
| | 000 | | 281 (-12.5°) |
| | 001 | | 213 (-14.4°) |
| | 010 | | 156 (-17.2°) |
| | 011 | | 407 (-10.4°) |
| | 100 | | 102 (-22.9°) |
| | 101 | | 58 (-35.7°) |
| | 110[1] | | 28.8 (-42.1°) |
| | 111[1] | | 14.4 (-61.7°) |

1. *For ODR ≥ 3840 Hz the cases LPF1_G_BW_[2:0] = 11x should be avoided due to low LPF1 roll-off at higher frequency.*

If the gyroscope is configured in low-power mode, the gyroscope filtering chain presented above is bypassed. The bandwidth in low-power mode is indicated in the following table.

**Table 24. Gyroscope low-power mode bandwidth**

| Gyroscope ODR [Hz] | Bandwidth [Hz] |
|---|---|
| 7.5 | 2.3 |
| 15 | 4.6 |
| 30 | 9.1 |
| 60 | 18 |
| 120 | 36 |
| 240 | 71 |

## 3.13 Gyroscope turn-on/off time

Turn-on/off time has to be considered also for the gyroscope sensor when switching its modes or when the gyroscope ODR is changed.

When the device is configured in mode 1/2, the maximum overall turn-on/off time in order to switch gyroscope power modes or gyroscope ODR is the one shown in Table 25. Gyroscope turn-on/off time in mode 1/2.

*Note: The gyroscope ODR timing is not impacted by power mode changes (the new configuration is effective after the completion of the current period).*

**Table 25. Gyroscope turn-on/off time in mode 1/2**

| Starting mode | Target mode | Max turn-on/off time(1) |
|---|---|---|
| Power-down | Sleep | 70 ms |
| Power-down | Low-power mode | 70 ms + discard 1 sample |
| Power-down | High-performance mode | 70 ms + Table 26 + Table 27(2) |
| Power-down | High-accuracy ODR mode | 70 ms + Table 26 + Table 27(2) |
| Sleep | Low-power mode | Discard 1 sample |
| Sleep | High-performance mode | Table 26 + Table 27(2) |
| Low-power mode | High-performance mode | Table 26 + Table 27(2) |
| Low-power mode | Low-power mode (ODR change) | Discard 1 sample |
| High-performance mode | Low-power mode | Discard 1 sample |
| High-performance mode | High-performance mode (ODR change) | Discard 2 samples |
| Low-power / high-performance / high-accuracy ODR mode | Power-down | 5 ms |

1. *Settling time @ 99% of the final value*
2. *Only when LPF1 is enabled*

**Table 26. Gyroscope samples to be discarded in mode 1/2 (LPF1 disabled)**

| Gyroscope ODR [Hz] | Number of samples to be discarded(1) (high-performance mode) | Number of samples to be discarded(1) (high-accuracy ODR mode) |
|---|---|---|
| 7.5 Hz | 2 | - |
| 15 Hz | 2 | 5 |
| 30 Hz | 2 | 5 |
| 60 Hz | 3 | 6 |
| 120 Hz | 3 | 6 |
| 240 Hz | 4 | 7 |
| 480 Hz | 5 | 8 |
| 960 Hz | 6 | 9 |
| 1920 Hz | 10 | 13 |
| 3840 Hz | 18 | 21 |
| 7680 Hz | 33 | 36 |

1. *Settling time @ 99% of the final value*

**Table 27. Gyroscope chain settling time in mode 1/2 (LPF1 enabled)**

| LPF1_G_BW_[2:0] | Maximum settling time @ each ODR [ms][(1)] |
|---|---|
| 000 | 8.2 |
| 001 | 13.0 |
| 010 | 18.7 |
| 011 | 5.2 |
| 100 | 29.6 |
| 101 | 53.1 |
| 110 | 48.2 |
| 111 | 96.7 |

1. *Settling time @ 99% of the final value*

# 4 Mode 1 - reading output data

## 4.1 Startup sequence

Once the device is powered up, it automatically downloads the calibration coefficients from the embedded flash to the internal registers. When the boot procedure is completed, that is, after 10 ms (maximum), the low-*g* accelerometer, gyroscope, and high-*g* accelerometer automatically enter power-down mode.

To turn on the low-*g* accelerometer and gather acceleration data through the I²C / MIPI I3C® / SPI interface, 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 low-*g* accelerometer:

1. Write INT1_CTRL = 01h // Low-*g* accelerometer data-ready interrupt on INT1
2. Write CTRL1 = 08h // ODR_XL = 480 Hz (high-performance mode)

To turn on the gyroscope and gather angular rate data through the I²C / MIPI I3C® / SPI interface, it is necessary to select one of the operating modes through the CTRL2 register.

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

1. Write INT1_CTRL = 02h // Gyroscope data-ready interrupt on INT1
2. Write CTRL2 = 08h // ODR_G = 480 Hz (high-performance mode)

To turn on the high-*g* accelerometer and gather acceleration with the high full scale over the I²C / MIPI I3C® / SPI interface, it is necessary to select one of the compatible operating modes for the low-*g* accelerometer through the CTRL1 register (Section 3.2: High-*g* accelerometer power modes and output data rate), then set the desired ODR and route the output to the UI_OUTx_x_A_HG (34h - 39h) registers through the CTRL1_XL_HG register. The following procedure can be used to configure the high-*g* accelerometer:

1. Write CTRL7 = 80h // High-*g* accelerometer data-ready interrupt on INT1
2. Write CTRL1_XL_HG = 98h // XL_HG_ REGOUT_EN = 1, ODR_XL_HG = 480 Hz

## 4.2 Using the status register

The device is provided with a STATUS_REG register, which can be polled to check when a new set of data is available. The XLDA bit is set to 1 when a new set of data is available in the low-*g* accelerometer output registers. The GDA bit is set to 1 when a new set of data is available in the gyroscope output registers. The XLHGDA bit is set to 1 when a new set of data is available in the high-*g* accelerometer output registers.

For the low-*g* accelerometer (the gyroscope and the high-*g* accelerometer are similar), the read of the output registers can be performed as follows:

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

## 4.3 Using the data-ready signal

The device can be configured to provide a hardware signal to indicate when a new set of measurement data is available to be read.

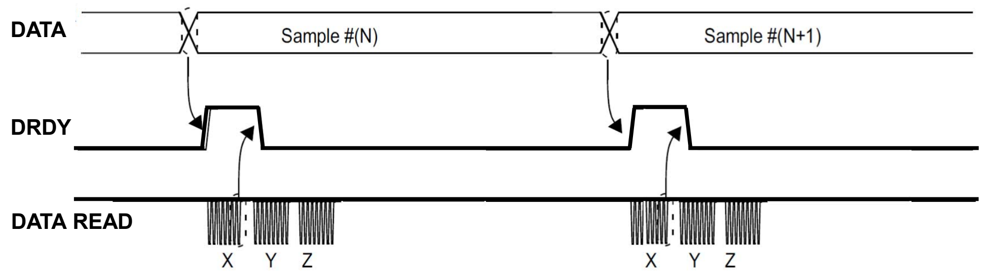
For the low-*g* accelerometer sensor, the data-ready signal is represented by the XLDA bit of the STATUS_REG register. The signal can be driven to the INT1 pin by setting the INT1_DRDY_XL bit of the INT1_CTRL register to 1 and to the INT2 pin by setting the INT2_DRDY_XL bit of the INT2_CTRL register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

For the gyroscope sensor, the data-ready signal is represented by the GDA bit of the STATUS_REG register. The signal can be driven to the INT1 pin by setting the INT1_DRDY_G bit of the INT1_CTRL register to 1 and to the INT2 pin by setting the INT2_DRDY_G bit of the INT2_CTRL register to 1.

For the high-*g* accelerometer sensor, the data-ready signal is represented by the XLHGDA bit of the STATUS_REG register. The signal can be routed to the INT1/INT2 pin by setting the INT1_DRDY_XL_HG/ INT2_DRDY_XL_HG bits of the CTRL7 register to 1.

The data-ready signal rises to 1 when a new set of data has been generated and it is available to be read. The data-ready signal can be either latched or pulsed. If the DRDY_PULSED bit of the CTRL4 register is set to 0 (default value), then the data-ready signal is latched and the interrupt is reset when the higher byte of one axis is read (29h, 2Bh, 2Dh registers for the low-*g* accelerometer; 23h, 25h, 27h registers for the gyroscope; 35h, 37h, 39h registers for the high-*g* accelerometer, when the XL_HG_REGOUT_EN bit of the CTRL1_XL_HG register is set to 1). If the DRDY_PULSED bit of the CTRL4 register is set to 1, then the data-ready is pulsed and the duration of the pulse observed on the interrupt pin is 65 µs. If the low-*g* accelerometer, the high-*g* accelerometer, or the gyroscope is configured in HAODR mode, the duration of the pulse observed on the interrupt pin is 43 µs. Pulsed mode is not applied to the XLDA, XLHGDA, and GDA bits, which are always latched.

**Figure 7. Data-ready signal**



### 4.3.1 DRDY mask functionality

Setting the DRDY_MASK bit of the CTRL4 register to 1, the low-*g* accelerometer, high-*g* accelerometer, and gyroscope data-ready signals are masked until the settling of the sensor filters is completed.

When FIFO is active and the DRDY_MASK bit is set to 1, low-*g* accelerometer, high-*g* accelerometer, and gyroscope invalid samples stored in FIFO can be equal to 7FFFh, 7FFEh, or 7FFDh. In this way, a tag is applied to the invalid samples stored in the FIFO buffer so that they can be easily identified and discarded during data postprocessing.

Referring to the low-*g* accelerometer UI chain, the DRDY mask functionality operates on all the power modes, full scales, and ODRs, considering also runtime changes. It covers the HP or LPF2 filter configuration up to ODR / 20 and it can be combined with the XL_FASTSETTL_MODE bit of the CTRL9 register for managing all the other filter configurations. If both the DRDY_MASK and the XL_FASTSETTL_MODE bits are set to 1, all the data-ready signals are masked until the internal filters are settled.

Referring to the high-*g* accelerometer chain, the DRDY mask functionality operates on all the power modes, full scales, and ODRs, considering also runtime changes.

Referring to the gyroscope UI chain, the DRDY mask functionality operates on all the power modes, full scales, and ODRs, considering also runtime changes.

The DRDY mask feature is not available for HAODR mode.

*Note:* *If the DRDY mask functionality is enabled, in order to guarantee the proper masking of the low-g accelerometer sensor data-ready signal until the settling of the accelerometer filtering chain is completed, it is necessary to wait at least 600 µs between two low-g accelerometer power-mode changes.*

## 4.4 Using the block data update (BDU) feature

If reading the low-*g*/high-*g* accelerometer/gyroscope data is not synchronized with either the XLDA/XLHGDA/GDA bits in the STATUS_REG register or with the data-ready signal driven to the INT1/INT2 pins, it is strongly recommended to set the BDU (block data update) bit to 1 in the CTRL3 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, OUTX_H_A(G) and OUTX_L_A(G), OUTY_H_A(G), and OUTY_L_A(G), OUTZ_H_A(G), and OUTZ_L_A(G)) is initiated, the refresh for that pair is blocked until both the MSB and LSB of the data are read. The same applies to the high-*g* accelerometer output data registers.

*Note: BDU only guarantees that the LSB and MSB 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 BDU feature also acts on the FIFO_STATUS1 and FIFO_STATUS2 registers. When the BDU bit is set to 1, it is mandatory to read FIFO_STATUS1 first and then FIFO_STATUS2.

## 4.5 Understanding output data

The measured acceleration data on the low-*g* accelerometer are sent to the OUTX_H_A, OUTX_L_A, OUTY_H_A, OUTY_L_A, OUTZ_H_A, and OUTZ_L_A 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 measured angular rate data are sent to the OUTX_H_G, OUTX_L_G, OUTY_H_G, OUTY_L_G, OUTZ_H_G, and OUTZ_L_G registers. These registers contain, respectively, the most significant part and the least significant part of the angular rate signals acting on the X, Y, and Z axes.

The measured acceleration data on the high-*g* accelerometer are sent to the UI_OUTX_H_A_HG, UI_OUTX_L_A_HG, UI_OUTY_H_A_HG, UI_OUTY_L_A_HG, UI_OUTZ_H_A_HG, and UI_OUTX_L_A_HG registers when the XL_HG_REGOUT_EN bit of the CTRL1_XL_HG register is set to 1. These registers contain, respectively, the most significant part and the least significant part of the acceleration signal 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_A(G) & OUTX_L_A(G), OUTY_H_A(G) & OUTY_L_A(G), OUTZ_H_A(G) & OUTZ_L_A(G) and it is expressed as a two's complement number. The same applies to the high-*g* accelerometer output data registers.

Low-*g* acceleration, angular rate, and high-*g* acceleration data are represented as 16-bit numbers. In order to translate them to their corresponding physical representation, a sensitivity parameter must be applied. This sensitivity value depends on the selected full-scale range (refer to the datasheet). In detail:

- Each low-*g* or high-*g* acceleration sample must be multiplied by the proper sensitivity parameter LA_So (linear acceleration sensitivity expressed in m*g*/LSB) in order to obtain the corresponding value in m*g*.
- Each angular rate sample must be multiplied by the proper sensitivity parameter G_So (angular rate sensitivity expressed in mdps/LSB) in order to obtain the corresponding value in mdps.

### 4.5.1 Examples of output data

Table 28. Content of output data registers vs. acceleration (FS_XL = ±2 *g*) provides a few basic examples of the low-*g* accelerometer data that is read from the data registers when the device is subjected to a given acceleration.

Table 29. Content of output data registers vs. acceleration (FS_HG_XL = ±32 *g*) provides a few basic examples of the high-*g* accelerometer data that is read from the data registers when the device is subjected to a given acceleration.

Table 30. Content of output data registers vs. angular rate (FS_G = ±250 dps) provides a few basic examples of the gyroscope data that is read from the data registers when the device is subjected to a given angular rate.

The values listed in the following tables are given under the hypothesis of perfect device calibration (that is, no offset, no gain error, and so on).

**Table 28. Content of output data registers vs. acceleration (FS_XL = ±2 *g*)**

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

**Table 29. Content of output data registers vs. acceleration (FS_HG_XL = ±32 *g*)**

| Acceleration values | Register address | |
|---|---|---|
| | UI_OUTX_H_A_HG (35h) | UI_OUTX_L_A_HG (34h) |
| 0 *g* | 00h | 00h |
| 350 m*g* | 01 | 66h |
| 1 *g* | 04 | 00h |
| -350 m*g* | FEh | 9Ah |
| -1 *g* | FC | 00h |

**Table 30. Content of output data registers vs. angular rate (FS_G = ±250 dps)**

| Angular rate values | Register address | |
|---|---|---|
| | OUTX_H_G (23h) | OUTX_L_G (22h) |
| 0 dps | 00h | 00h |
| 100 dps | 2Ch | A4h |
| 200 dps | 59h | 49h |
| -100 dps | D3h | 5Ch |
| -200 dps | A6h | B7h |

## 4.6 Low-*g* accelerometer offset registers

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

The low-*g* accelerometer offset block can be enabled by setting the USR_OFF_ON_OUT bit of the CTRL9 register. The offset value set in the offset registers is internally added to the measured acceleration value for the respective axis. Internally processed data are then sent to the low-*g* accelerometer output register and to the FIFO (if enabled). These register values are expressed as an 8-bit word in two's complement and must be in the range [-127, 127].

The weight [*g*/LSB] to be applied to the offset register values is independent of the low-*g* accelerometer selected full scale and can be configured using the USR_OFF_W bit of the CTRL9 register:

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

## 4.7 High-*g* accelerometer offset registers

The device provides high-*g* accelerometer offset registers (XL_HG_X_OFS_USR, XL_HG_Y_OFS_USR, XL_HG_Z_OFS_USR) that are used to apply an offset to the high-*g* accelerometer output data.

The high-*g* accelerometer offset block is enabled by setting the HG_USR_OFF_ON_OUT bit in the CTRL1_XL_HG register.

The offset value set in the offset registers is internally added to the measured acceleration value for the respective axis. Internally processed data are then sent to the high-*g* accelerometer output register and to the FIFO (if enabled). These register values are expressed as an 8-bit word in two's complement and must be in the range [-127, 127]. The weight [*g*/LSB] of the high-*g* accelerometer is full-scale independent and is always set to 0.25 *g*/LSB (not configurable).

# 5 Interrupt generation

Interrupt generation is based on the low-*g* accelerometer data only, so, for interrupt-generation purposes, the low-*g* accelerometer sensor has to be set in an active operating mode (not in power-down). The high-*g* accelerometer and gyroscope sensors can be configured in power-down mode since they are not involved in interrupt generation.

The interrupt generator can be configured to detect:

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

An additional interrupt generation block is based on high-*g* accelerometer data. In order to use it, the high-*g* accelerometer sensor has to be set in an active operating mode (not in power-down). The low-*g* and gyroscope sensors can be configured in power-down mode since they are not involved in interrupt generation.

The high-*g* interrupt generation can be configured to detect wake-up and shock events.

The device can also efficiently run the sensor-related features specified in Android, saving power and enabling faster reaction time. The following functions are implemented in the hardware:

- Significant motion
- Relative tilt
- Pedometer functions
- Timestamp
- Sensor fusion functions (game rotation vector, gravity vector, gyroscope bias)

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 note available on www.st.com.

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

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 (default behavior). As an additional feature, by setting the INT_EN_I3C bit of the CTRL5 register to 1, the interrupt pins are activated even if using the MIPI I3C® interface.

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

## 5.1 Interrupt pin configuration

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

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

**Table 31. INT1_CTRL register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | INT1_CNT_BDR | INT1_FIFO_FULL | INT1_FIFO_OVR | INT1_FIFO_TH | 0 | INT1_DRDY_G | INT1_DRDY_XL |

- INT1_CNT_BDR: FIFO COUNTER_BDR_IA interrupt on INT1
- INT1_FIFO_FULL: FIFO full flag interrupt on INT1
- INT1_FIFO_OVR: FIFO overrun flag interrupt on INT1
- INT1_FIFO_TH: FIFO threshold interrupt on INT1
- INT1_DRDY_G: gyroscope data-ready on INT1
- INT1_DRDY_XL: low-*g* accelerometer data-ready on INT1

**Table 32. MD1_CFG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| INT1_SLEEP_CHANGE | INT1_SINGLE_TAP | INT1_WU | INT1_FF | INT1_DOUBLE_TAP | INT1_6D | INT1_EMB_FUNC | INT1_SHUB |

- INT1_SLEEP_CHANGE: activity/inactivity recognition event interrupt on INT1
- INT1_SINGLE_TAP: single-tap interrupt on INT1
- INT1_WU: wake-up interrupt on INT1
- INT1_FF: free-fall interrupt on INT1
- INT1_DOUBLE_TAP: double-tap interrupt on INT1
- INT1_6D: 6D detection interrupt on INT1
- INT1_EMB_FUNC: embedded functions interrupt on INT1 (refer to Section 6: Embedded functions for more details)
- INT1_SHUB: sensor hub end operation interrupt on INT1

**Table 33. INT2_CTRL register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| INT2_EMB_FUNC_ENDOP | INT2_CNT_BDR | INT2_FIFO_FULL | INT2_FIFO_OVR | INT2_FIFO_TH | 0 | INT2_DRDY_G | INT2_DRDY_XL |

- INT2_EMB_FUNC_ENDOP: embedded functions end of operations interrupt on INT2. This bit is intended to be used for debugging purposes. For this reason, it is not recommended to enable it if other interrupt signals are intended to be routed to the INT2 pin. When it is enabled, the INT2 pin is set to high level if no embedded function is running, otherwise, it is set to low level if any embedded function is running. For this reason, it can be used to measure the execution time of the embedded functions.
- INT2_CNT_BDR: FIFO COUNTER_BDR_IA interrupt on INT2
- INT2_FIFO_FULL: FIFO full flag interrupt on INT2
- INT2_FIFO_OVR: FIFO overrun flag interrupt on INT2
- INT2_FIFO_TH: FIFO threshold interrupt on INT2
- INT2_DRDY_G: gyroscope data-ready on INT2
- INT2_DRDY_XL: low-*g* accelerometer data-ready on INT2

The INT2_DRDY_TEMP bit of the CTRL4 register enables the temperature data-ready interrupt on the INT2 pin.

Table 34. **MD2_CFG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| INT2_SLEEP _CHANGE | INT2_ SINGLE_TAP | INT2_WU | INT2_FF | INT2_ DOUBLE_TAP | INT2_6D | INT2_ EMB_FUNC | INT2_ TIMESTAMP |

- INT2_SLEEP_CHANGE: activity/inactivity recognition event interrupt on INT2
- INT2_SINGLE_TAP: single-tap interrupt on INT2
- INT2_WU: wake-up interrupt on INT2
- INT2_FF: free-fall interrupt on INT2
- INT2_DOUBLE_TAP: double-tap interrupt on INT2
- INT2_6D: 6D detection interrupt on INT2
- INT2_EMB_FUNC: embedded functions interrupt on INT2 (refer to Section 6: Embedded functions for more details)
- INT2_TIMESTAMP: timestamp overflow alert interrupt on INT2

If multiple interrupt signals are routed to the same interrupt 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:

- WAKE_UP_SRC, TAP_SRC, D6D_SRC (basic interrupt functions)
- STATUS_REG (for data-ready signals)
- EMB_FUNC_STATUS_MAINPAGE / EMB_FUNC_STATUS (for embedded functions)
- FSM_STATUS_MAINPAGE / FSM_STATUS (for finite state machine)
- MLC_STATUS_MAINPAGE / MLC_STATUS (for machine learning core)
- STATUS_CONTROLLER_MAINPAGE / STATUS_CONTROLLER (for sensor hub)
- FIFO_STATUS2 (for FIFO)

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

The INT2_on_INT1 bit of the CTRL4 register allows driving some specific interrupt signals in logic “OR” on the INT1 pin (by setting this bit to 1). When this bit is set to 0, the interrupt signals are divided between the INT1 and INT2 pins. When this bit is set to 1, the movable interrupts are and INT2_EMB_FUNC_ENDOP (enabled through the INT2_CTRL register), INT2_TIMESTAMP (enabled through the MD2_CFG register), INT2_DRDY_TEMP (enabled through the CTRL4 register).

The low-*g* accelerometer basic interrupts have to be enabled by setting the INTERRUPTS_ENABLE bit in the FUNCTIONS_ENABLE register.

The LIR bit of the TAP_CFG0 register enables the latched interrupt for the basic interrupt functions: when this bit is set to 1 and the interrupt flag is sent to the INT1 pin and/or INT2 pin, the interrupt remains active until the ALL_INT_SRC register or the corresponding source register is read. 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. The DIS_RST_LIR_ALL_INT bit of the FUNCTIONS_ENABLE register can be set to 1 in order to avoid resetting the latched interrupt signals by reading the ALL_INT_SRC register. This feature is useful in order to not reset some status flags before reading the corresponding status register.

## 5.2 High-*g* interrupt pin configuration

Both interrupt pins can be used to route the interrupt signals related to the high-*g* accelerometer features. The functionality of these pins is selected through the dedicated bits in the CTRL7, HG_FUNCTIONS_ENABLE, and INACTIVITY_THS registers for both INT1 / INT2 pins.

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 corresponding bit has to be set to 1.

**Table 35. CTRL7 register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| INT1_DRDY_XL_HG | INT2_DRDY_XL_HG | 0 | 0 | 0 | 0 | 0 | LPF1_G_EN |

- INT1_DRDY_XL_HG: high-*g* accelerometer data-ready on INT1
- INT2_DRDY_XL_HG: high-*g* accelerometer data-ready on INT2

**Table 36. HG_FUNCTIONS_ENABLE register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| HG_INTERRUPTS_ENABLE | HG_WU_CHANGE_INT_SEL | INT2_HG_WU | INT1_HG_WU | HG_SHOCK_DUR_3 | HG_SHOCK_DUR_2 | HG_SHOCK_DUR_1 | HG_SHOCK_DUR_0 |

- HG_INTERRUPTS_ENABLE: enables / disables the high-*g* accelerometer wake-up / shock detection interrupt generator
- HG_WU_CHANGE_INT_SEL: selects the type of wake-up interrupt to be sent to the INT1 / INT2 pin
- INT2_HG_WU: high-*g* accelerometer wake-up interrupt on INT2
- INT1_HG_WU: high-*g* accelerometer wake-up interrupt on INT1

**Table 37. INACTIVITY_THS register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| INT2_HG_SHOCK_CHANGE | INT1_HG_SHOCK_CHANGE | INACT_THS_5 | INACT_THS_4 | INACT THS_3 | INACT_THS_2 | INACT THS_1 | INACT_THS_0 |

- INT2_HG_SHOCK_CHANGE: high-*g* accelerometer shock interrupt on INT2
- INT1_HG_SHOCK_CHANGE: high-*g* accelerometer shock interrupt on INT2

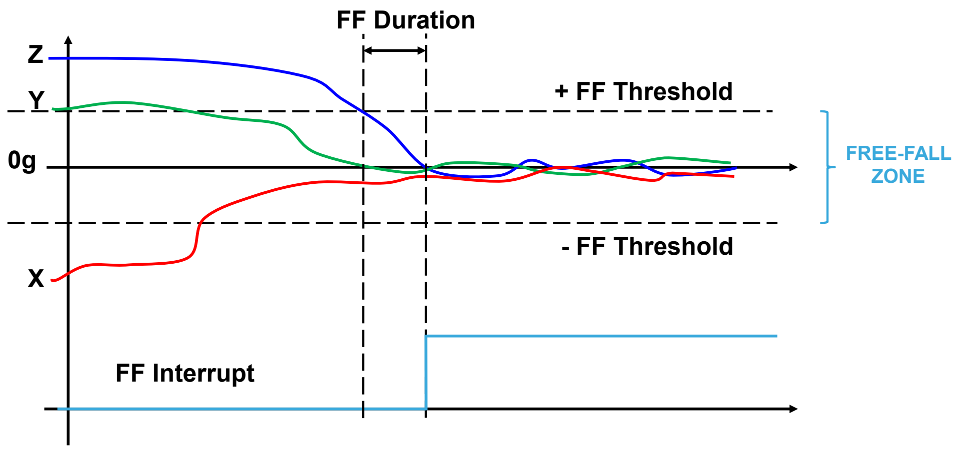
The high-*g* accelerometer basic interrupts have to be enabled by setting the HG_INTERRUPTS_ENABLE bit in the HG_FUNCTIONS_ENABLE register.

The related interrupt status bits can be found in the HG_WAKE_UP_SRC register, and the OR combination of the high-*g* accelerometer wake-up and shock detection feature can be monitored through the HG_IA bit in the ALL_INT_SRC register. Refer to Section 5.8: High-*g* wake-up and shock detection for a detailed explanation of the high-*g* accelerometer wake-up / shock detection interrupts behavior.

## 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 of the low-*g* accelerometer 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 with 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 8).

**Figure 8. Free-fall interrupt**




The free-fall interrupt signal can be enabled by setting the INTERRUPTS_ENABLE bit in the FUNCTIONS_ENABLE register to 1 and can be driven to the two interrupt pins by setting the INT1_FF bit of the MD1_CFG register to 1 or the INT2_FF bit of the MD2_CFG register to 1. It can also be checked by reading the FF_IA bit of the WAKE_UP_SRC or ALL_INT_SRC register.

If latched mode is disabled (LIR bit of TAP_CFG0 is set to 0), the interrupt signal is automatically reset when the free-fall condition is no longer verified. If latched 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 or ALL_INT_SRC register. If latched mode is enabled but the interrupt signal is not driven to the interrupt pins, the latch feature does not take effect.

The FREE_FALL register is used to configure the threshold parameter. The unsigned threshold value is related to the value of the FF_THS_[2:0] field value as indicated in Table 38. Free-fall threshold LSB value. The values given in this table are valid for each low-*g* accelerometer full-scale value.

**Table 38. Free-fall threshold LSB value**

| FREE_FALL - FF_THS_[2:0] | Threshold LSB value [m*g*] |
|---|---|
| 000 | 156 |
| 001 | 219 |
| 010 | 250 |
| 011 | 312 |
| 100 | 344 |
| 101 | 406 |
| 110 | 469 |
| 111 | 500 |

Duration time is measured in N/ODR_XL, where N is the content of the FF_DUR_[5:0] field of the FREE_FALL / WAKE_UP_DUR registers and ODR_XL is the low-*g* accelerometer data rate.

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

| | | |
|---|---|---|
| 1. | Write 08h to CTRL1 | // Turn on the low-*g* accelerometer (ODR = 480 Hz) |
| 2. | Write 01h to TAP_CFG0 | // Enable latched mode |
| 3. | Write 80h to FUNCTIONS_ENABLE | // Enable interrupt functions |
| 4. | Write 00h to WAKE_UP_DUR | // Set event duration (FF_DUR_5 bit) |
| 5. | Write 33h to FREE_FALL | // Set FF threshold (FF_THS_[2:0] = 011) |
| | | // Set six samples event duration (FF_DUR_[5:0] = 000110) |
| 6. | Write 10h to MD1_CFG | // FF interrupt driven to INT1 pin |

The sample code sets the threshold 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_XL = 6/480 Hz ~= 12.5 msec in order to avoid false detections.

## 5.4 Wake-up interrupt

The wake-up feature can be implemented using either the slope filter (see Section 3.8.1: Low-*g* accelerometer slope filter for more details) or the high-pass digital filter, as illustrated in Figure 3. Low-*g* accelerometer filtering chain (UI path). The filter to be applied can be selected using the SLOPE_FDS bit of the TAP_CFG0 register. If this bit is set to 0 (default value), the slope filter is used; if it is set to 1, the HPF digital filter is used. Moreover, it is possible to configure the wake-up feature as an absolute wake-up with respect to a programmable position. This can be done by setting both the SLOPE_FDS bit of the TAP_CFG0 register and the USR_OFF_ON_WU bit of the WAKE_UP_THS register to 1. Using this configuration, the input data for the wake-up function comes from the low-pass filter path and the programmable position is added as an offset. The programmable position can be configured through the X_OFS_USR, Y_OFS_USR and Z_OFS_USR registers (refer to Section 4.6: Low-*g* accelerometer offset registers for more details).

The wake-up interrupt signal is generated if a certain number of consecutive filtered data exceed the configured threshold (Figure 9. Wake-up interrupt (using the slope filter)).

The unsigned threshold value is defined using the WK_THS_[5:0] bits of the WAKE_UP_THS register. The value of 1 LSB of these 6 bits depends on the value of the WU_INACT_THS_W_[2:0] bits of the INACTIVITY_DUR register as shown in the table below.

Table 39. **Wake-up threshold resolution**

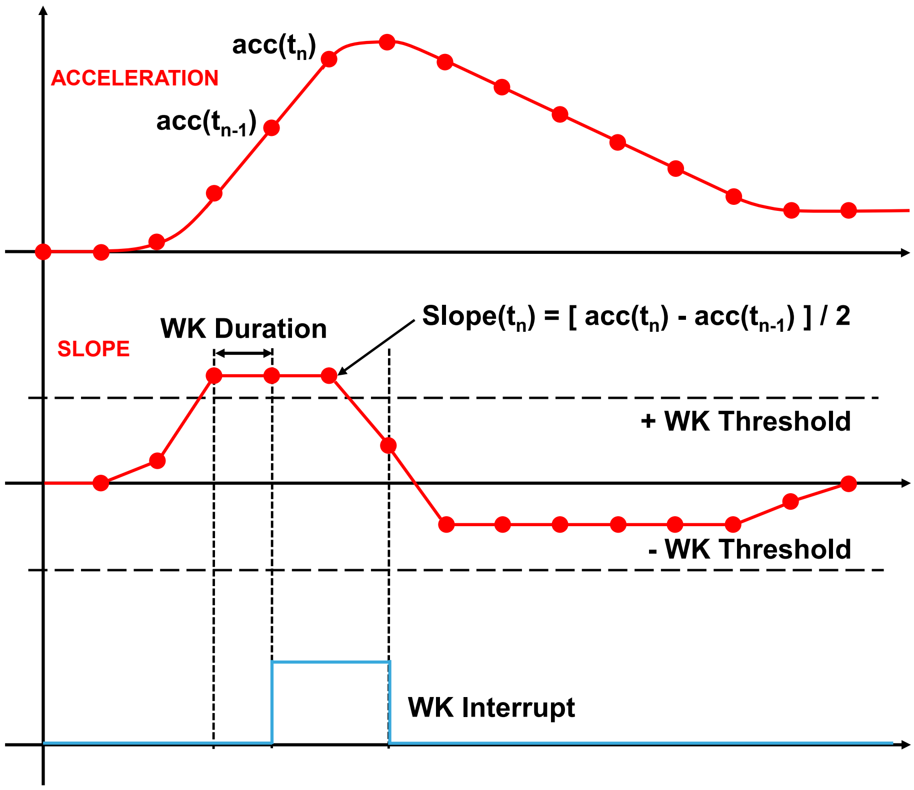
| WU_INACT_THS_W_[2:0] | 1 LSB resolution |
|---|---|
| 000 | 7.8125 m*g* |
| 001 | 15.625 m*g* |
| 010 | 31.25 m*g* |
| 011 | 62.5 m*g* |
| 100 | 125 m*g* |
| 101 | 250 m*g* |
| 110 | 250 m*g* |
| 111 | 250 m*g* |

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

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

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

**Figure 9. Wake-up interrupt (using the slope filter)**




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

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

1. Write 34h to INACTIVITY_DUR // Set wake-up threshold resolution to 62.5 m*g*
2. Write 11h to TAP_CFG0 // Select HPF path and enable latched mode
3. Write 01h to WAKE_UP_THS // Set wake-up threshold
4. Write 00h to WAKE_UP_DUR // Set duration to 0
5. Write 20h to MD1_CFG // Wake-up interrupt driven to INT1 pin
6. Write 80h to FUNCTIONS_ENABLE // Enable interrupt functions
7. Write 08h to CTRL1 // Turn on the low-*g* accelerometer (ODR = 480 Hz)

Since the duration time is set to zero, the wake-up interrupt signal is generated for each X, Y, Z filtered data exceeding the configured threshold. The WK_THS field of the WAKE_UP_THS register is set to 000001 and the resolution of 1 LSB is set to 62.5 m*g* (WU_INACT_THS_W_[2:0] bits of INACTIVITY_DUR register are set to 011), therefore the wake-up threshold is 62.5 m*g*.

Since the wake-up functionality is implemented using the slope/high-pass digital filter, it is necessary to consider the settling time of the filter just after this functionality is enabled. For example, when using the slope filter (but a similar consideration can be done for the high-pass digital filter usage) the wake-up functionality is based on the comparison of the threshold value with half of the difference of the acceleration of the current (X, Y, Z) sample and the previous one (refer to Section 3.8.1: Low-*g* accelerometer slope filter).

At the very first sample, the slope filter output is calculated as half of the difference of the current sample, for example (X, Y, Z) = (0, 0, 1) *g*, with the previous one, which is (X, Y, Z) = (0, 0, 0) *g* since no sample has been generated yet. For this reason, on the Z-axis, the first output value of the slope filter is (1 - 0) / 2 = 0.5 *g* = 500 m*g* and it could be higher than the threshold value in which case a spurious interrupt event is generated. The interrupt signal is kept high for 1 ODR then it goes low.

In order to avoid the spurious interrupt generation due to the settling of the digital slope / high-pass filter, it is possible to mask the execution trigger of the basic interrupt functions during the digital filter settling by configuring to 1 both the XL_FASTSETTL_MODE bit of the CTRL9 register and the HW_FUNC_MASK_XL_SETTL of the TAP_CFG0 register.

The wake-up configuration procedure described above can be easily modified as follows:

1. Write 20h to CTRL9 // Set XL_FASTSETTL_MODE = 1
2. Write 34h to INACTIVITY_DUR // Set wake-up threshold resolution to 62.5 m*g*
3. Write 31h to TAP_CFG0 // Set HW_FUNC_MASK_XL_SETTL = 1, select HPF path and enable latched mode
4. Write 01h to WAKE_UP_THS // Set wake-up threshold
5. Write 00h to WAKE_UP_DUR // Set duration to 0
6. Write 20h to MD1_CFG // Wake-up interrupt driven to INT1 pin
7. Write 80h to FUNCTIONS_ENABLE // Enable interrupt functions
8. Write 08h to CTRL1 // Turn on the low-*g* accelerometer (ODR = 480 Hz)

## 5.5 6D/4D orientation detection

The low-*g* accelerometer provides the capability to detect the orientation of the device in space, enabling easy implementation of energy-saving procedures and automatic screen rotation for mobile devices.

### 5.5.1 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 reasserted 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 D6D_SRC register indicate which axis has triggered the 6D event.

In more detail:

**Table 40. D6D_SRC register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | D6D_IA | ZH | ZL | YH | YL | XH | XL |

- D6D_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 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 SIXD_THS_[1:0] bits of the TAP_THS_6D register are used to select the threshold value used to detect the change in device orientation. The threshold values given in the following table are valid for each low-*g* accelerometer full-scale value.

**Table 41. Threshold for 4D/6D function**

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

The low-pass filter LPF2 can also be used in 6D functionality by setting the LOW_PASS_ON_6D bit of the TAP_CFG0 register to 1.

This interrupt signal can be enabled by setting the INTERRUPTS_ENABLE bit in the FUNCTIONS_ENABLE register to 1 and can be driven to the two interrupt pins by setting the INT1_6D bit of the MD1_CFG register or the INT2_6D bit of the MD2_CFG register to 1. It can also be checked by reading the D6D_IA bit of the D6D_SRC or ALL_INT_SRC register.

If latched mode is disabled (LIR bit of TAP_CFG is set to 0), the interrupt signal is active only for 1/ODR_XL then it is automatically disserted (ODR_XL is the low-*g* accelerometer output data rate). If latched 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 read of the D6D_SRC or 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 LIR configuration. They correspond to the current state of the device when the D6D_SRC register is read. If latched 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 10. 6D recognized orientations, the content of the D6D_SRC register for each position is shown in Table 42. D6D_SRC register in 6D positions.

**Figure 10.** 6D recognized orientations

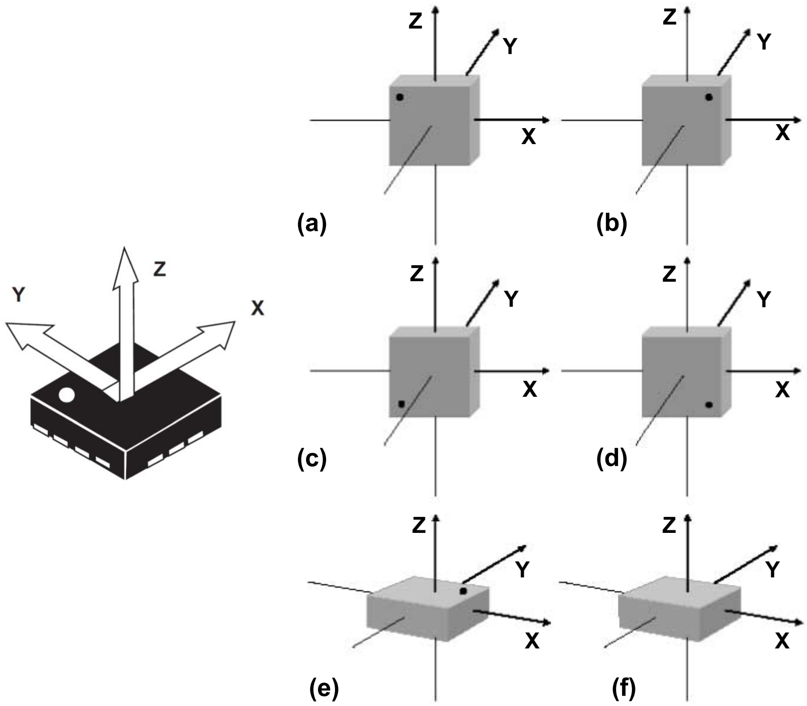



**Table 42.** D6D_SRC register in 6D positions

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

A basic software routine for 6D orientation detection is as follows.

1. Write 41h to TAP_CFG0 // Enable LPF2 filter for 6D functionality and latched mode
2. Write 40h to TAP_THS_6D // Set 6D threshold (SIXD_THS_[1:0] = 10 = 60 degrees)
3. Write 04h to MD1_CFG // 6D interrupt driven to INT1 pin
4. Write 80h to FUNCTIONS_ENABLE // Enable interrupt functions
5. Write 08h to CTRL1 // Turn on the low-*g* accelerometer (ODR = 480 Hz)

### 5.5.2 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 detection. It can be enabled by setting the D4D_EN bit of the TAP_THS_6D 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 42. D6D_SRC register in 6D positions.

## 5.6 Single-tap and double-tap recognition

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

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

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

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

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

Single and double-tap recognition work based on the selected output data rate. The recommended minimum low-*g* accelerometer ODR for these functions is 480 Hz.

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

### 5.6.1 Single tap

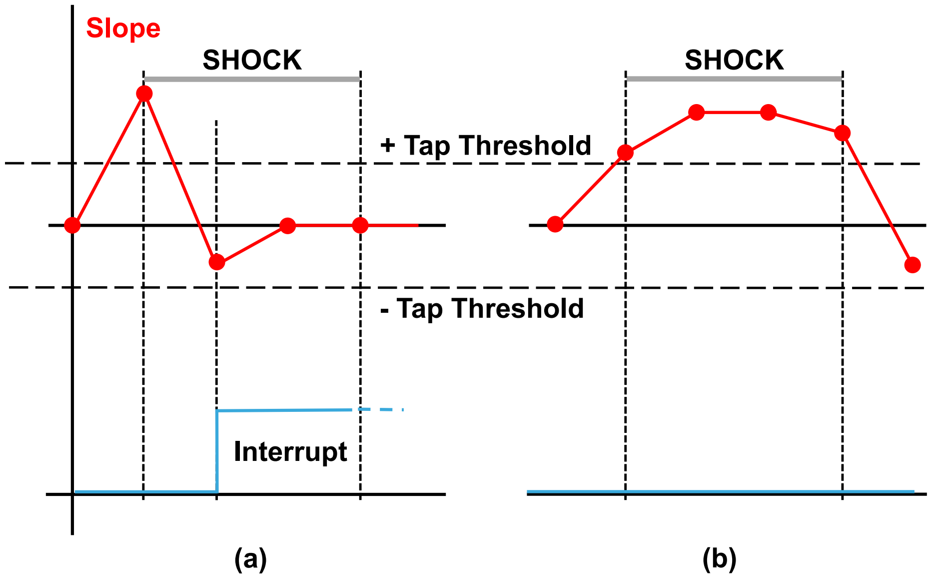
If the device is configured for single-tap event detection, an interrupt is generated when the slope data of the selected axis exceeds the programmed threshold, and returns below it within the shock time window.

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

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

In case (a) of Figure 11. Single-tap event recognition the single-tap event has been recognized, while in case (b) the tap has not been recognized because the slope data falls below the threshold after the shock time window has expired.

**Figure 11. Single-tap event recognition**



### 5.6.2 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 quiet and the duration time windows.

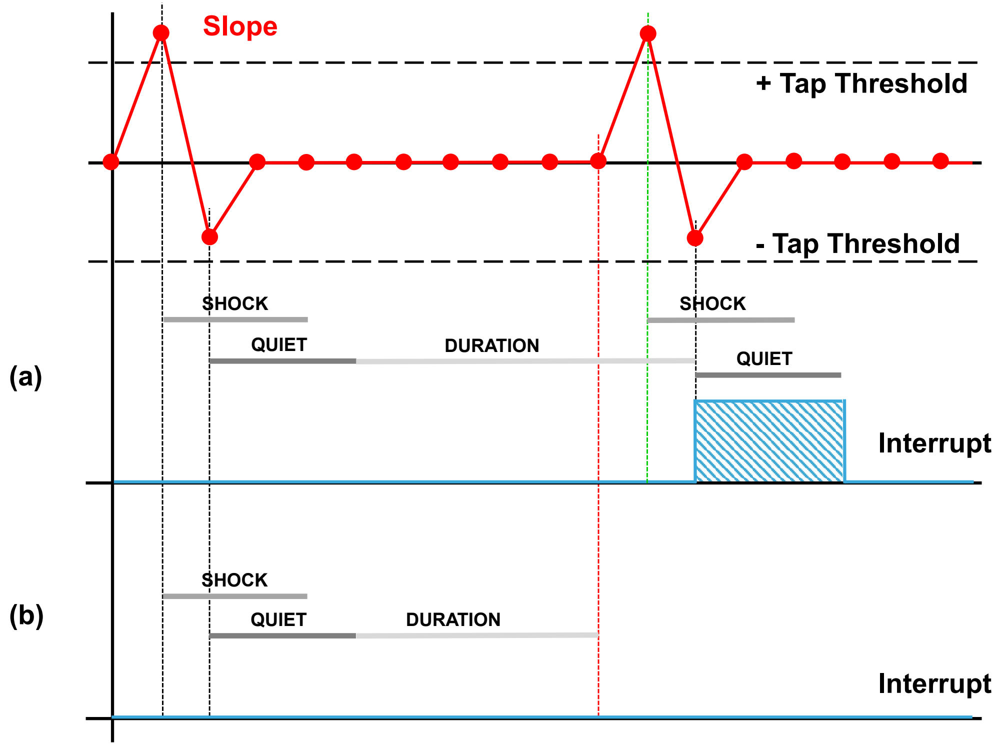
In particular, after the first tap has been recognized, the second tap detection procedure is delayed for an interval defined by the quiet time. This means that after the first tap has been recognized, the second tap detection procedure starts only if the slope data exceeds the threshold after the quiet window but before the duration window has expired. In case (a) of Figure 12, a double-tap event has been correctly recognized, while in case (b) the interrupt has not been generated because the slope data exceeds the threshold after the window interval has expired.

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

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

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

**Figure 12. Double-tap event recognition (LIR bit = 0)**



### 5.6.3 Single-tap and double-tap recognition configuration

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

Configurable parameters for tap recognition functionality are the tap thresholds (each axis has a dedicated threshold) and the shock, quiet, and duration time windows.

The TAP_THS_X_[4:0] bits of the TAP_CFG1 register, the TAP_THS_Y_[4:0] bits of the TAP_CFG2 register and the TAP_THS_Z_[4:0] bits of the TAP_THS_6D register are used to select the unsigned threshold value used to detect the tap event on the respective axis. The value of 1 LSB of these 5 bits depends on the selected low-*g* accelerometer full scale: 1 LSB = FS_XL / $2^5$. The unsigned threshold is applied to both positive and negative slope data.

Both single-tap and double-tap recognition functions apply to only one axis. If more than one axis are enabled and they are over the respective threshold, the algorithm continues to evaluate only the axis with highest priority. The priority can be configured through the TAP_PRIORITY_[2:0] bits of TAP_CFG1. The following table shows all the possible configurations.

**Table 43. TAP_PRIORITY_[2:0] bits configuration**

| TAP_PRIORITY_[2:0] | Maximum priority | Middle priority | Minimum priority |
|---|---|---|---|
| 000 | X | Y | Z |
| 001 | Y | X | Z |
| 010 | X | Z | Y |
| 011 | Z | Y | X |
| 100 | X | Y | Z |
| 101 | Y | Z | X |
| 110 | Z | X | Y |
| 111 | Z | Y | X |

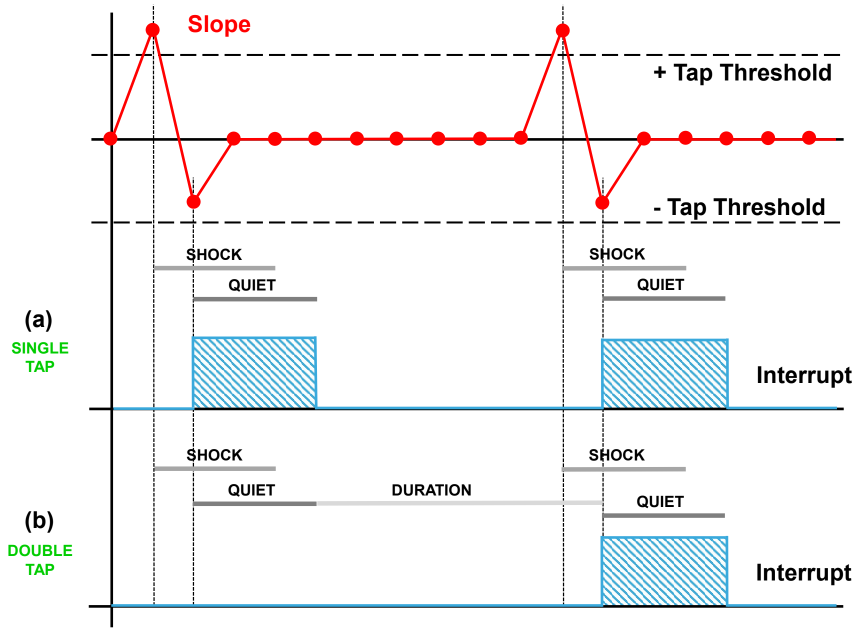
The shock time window defines the maximum duration of the overcoming threshold event: the acceleration must return below the threshold before the shock window has expired, otherwise the tap event is not detected. The SHOCK_[1:0] bits of the TAP_DUR register are used to set the shock time window value: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

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

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

Figure 13. Single and double-tap recognition (LIR bit = 0) illustrates a single-tap event (a) and a double-tap event (b). These interrupt signals can be driven to the two interrupt pins by setting the INT1_SINGLE_TAP bit of the MD1_CFG register or the INT2_SINGLE_TAP bit of the MD2_CFG register to 1 for the single-tap case, and setting the INT1_DOUBLE_TAP bit of the MD1_CFG register or the INT2_DOUBLE_TAP bit of the MD2_CFG register to 1 for the double-tap case.

Figure 13. **Single and double-tap recognition (LIR bit = 0)**




Tap interrupt signals can also be checked by reading the TAP_SRC (46h) register, described in the following table.

Table 44. **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 the case of a 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 the WAKE_UP_THS register to 0, only the single-tap recognition is enabled: double-tap recognition is disabled and cannot be detected. When the SINGLE_DOUBLE_TAP is set to 1, both single and double-tap recognition are enabled.

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

### 5.6.4 Single-tap example

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

| | | |
|---|---|---|
| 1. | Write 02h to TAP_CFG0 | // Enable tap detection on Z-axis |
| 2. | Write 00h to TAP_CFG1 | // Set X-axis threshold and axes priority |
| 3. | Write 00h to TAP_CFG2 | // Set Y-axis threshold |
| 4. | Write 02h to TAP_THS_6D | // Set Z-axis threshold |
| 5. | Write 06h to TAP_DUR | // Set quiet and shock time windows |
| 6. | Write 00h to WAKE_UP_THS | // Only single-tap enabled (SINGLE_DOUBLE_TAP = 0) |
| 7. | Write 80h to FUNCTIONS_ENABLE | // Enable hardware functions |
| 8. | Write 40h to MD1_CFG | // Single-tap interrupt driven to INT1 pin |
| 9. | Write 02h to CTRL8 | // FS_XL = ±8 *g* |
| 10. | Write 08h to CTRL1 | // Turn on the low-*g* accelerometer (480 Hz) |

In this example the TAP_THS_Z_[4:0] bits are set to 00010, therefore the tap threshold for the Z-axis is 500 m*g* (= 2 * FS_XL / $2^5$).

The SHOCK field of the TAP_DUR register is set to 10. An interrupt is generated when the slope data exceeds the programmed threshold, and returns below it within 33.3 ms (= 2 * 8 / ODR_XL) corresponding to the shock time window.

The QUIET field of the TAP_DUR register is set to 01. Since latched mode is disabled, the interrupt is kept high for the duration of the quiet window, therefore 8.3 ms (= 1 * 4 / ODR_XL).

### 5.6.5 Double-tap example

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

| | | |
|---|---|---|
| 1. | Write 02h to TAP_CFG0 | // Enable tap detection on Z-axis |
| 2. | Write 00h to TAP_CFG1 | // Set X-axis threshold and axes priority |
| 3. | Write 00h to TAP_CFG2 | // Set Y-axis threshold |
| 4. | Write 03h to TAP_THS_6D | // Set Z-axis threshold |
| 5. | Write 7Fh to TAP_DUR | // Set quiet and shock time windows |
| 6. | Write 80h to WAKE_UP_THS | // Single-tap and double-tap enabled (SINGLE_DOUBLE_TAP = 1) |
| 7. | Write 80h to FUNCTIONS_ENABLE | // Enable hardware functions |
| 8. | Write 08h to MD1_CFG | // Single-tap interrupt driven to INT1 pin |
| 9. | Write 02h to CTRL8 | // FS_XL = ±8 *g* |
| 10. | Write 08h to CTRL1 | // Turn on the low-*g* accelerometer (480 Hz) |

In this example the TAP_THS_Z_[4:0] bits are set to 00011, therefore the tap threshold is 750 m*g* (3 * FS_XL / $2^5$).

For interrupt generation, during the first and the second tap the slope data must return below the threshold before the shock window has expired. The SHOCK field of the TAP_DUR register is set to 11, therefore the shock time is 50 ms (= 3 * 8 / ODR_XL).

For interrupt generation, after the first tap recognition there must not be any slope data overthreshold during the quiet time window. Furthermore, since latched mode is disabled, the interrupt is kept high for the duration of the quiet window. The QUIET field of the TAP_DUR register is set to 11, therefore the quiet time is 25 ms (= 3 * 4 / ODR_XL).

For the maximum time between two consecutive detected taps, the DUR field of the TAP_DUR register is set to 0111, therefore the duration time is 533.3 ms (= 8 * 32 / ODR_XL).

## 5.7 Activity/inactivity and motion/stationary recognition

The working principle of activity/inactivity and motion/stationary embedded functions is similar to wake-up. If no movement condition is detected for a programmable time, an inactivity/stationary condition event is generated. Otherwise, when the low-*g* accelerometer data exceed the configurable threshold, an activity/motion condition event is generated.

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

When the activity/inactivity recognition function is activated, the device is able to automatically switch the low-*g* accelerometer power mode to low-power mode 1 and change the sampling rate to a configurable low ODR (available selectable ODRs are 1.875 Hz, 15 Hz, 30 Hz, 60 Hz) when the inactivity state is detected, while it is able to automatically switch back to the power mode and sampling rate selected through the OP_MODE_XL_[2:0] bits and ODR_XL_[3:0] bits of the CTRL1 register when the activity state is detected.

The target low-*g* accelerometer ODR for the inactivity state can be selected through the XL_INACT_ODR_[1:0] bits of the INACTIVITY_DUR register, with the values indicated in the table below.

**Table 45. Target low-*g* accelerometer ODR configuration for inactivity event**

| XL_INACT_ODR_[1:0] | ODR [Hz] |
|---|---|
| 00 | 1.875 |
| 01 | 15 |
| 10 | 30 |
| 11 | 60 |

This feature can be extended to the gyroscope, with three possible options:

- Gyroscope configurations do not change.
- Gyroscope enters sleep mode.
- Gyroscope enters power-down 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.

The activity/inactivity recognition function is enabled by setting the INTERRUPTS_ENABLE bit to 1 and configuring the INACT_EN_[1:0] bits of the FUNCTIONS_ENABLE register. If the INACT_EN_[1:0] bits of the FUNCTIONS_ENABLE register are equal to 00, the motion/stationary embedded function is enabled. Possible configurations of the inactivity event are summarized in the following table.

**Table 46. Inactivity event configuration**

| INACT_EN[1:0] | Low-*g* accelerometer | Gyroscope |
|---|---|---|
| 00 | Inactivity event disabled | Inactivity event disabled |
| 01 | Low-*g* accelerometer ODR set with XL_INACT_ODR_[1:0] bits (low-power mode 1) | Gyroscope configuration unchanged |
| 10 | Low-*g* accelerometer ODR set with XL_INACT_ODR_[1:0] bits (low-power mode 1) | Gyroscope in sleep mode |
| 11 | Low-*g* accelerometer ODR set with XL_INACT_ODR_[1:0] bits (low-power mode 1) | Gyroscope in power-down mode |

The activity/inactivity and motion/stationary recognition functions can be implemented using either the slope filter (see Section 3.8.1: Low-*g* accelerometer slope filter for more details) or the high-pass digital filter, as illustrated in Figure 3. Low-*g* accelerometer filtering chain (UI path). The filter to be applied can be selected using the SLOPE_FDS bit of the TAP_CFG0 register. If this bit is set to 0 (default value), the slope filter is used; if it is set to 1, the high-pass digital filter is used.

This function can be fully programmed by the user in terms of expected amplitude and timing of the filtered data by means of a dedicated set of registers (Figure 14. Activity/inactivity recognition (using the slope filter)).

The unsigned threshold value is defined using the INACT_THS_[5:0] bits of the INACTIVITY_THS register. The value of 1 LSB of these 6 bits depends on the value of the WU_INACT_THS_W_[2:0] bits of the INACTIVITY_DUR register as shown in the following table.

Table 47. **Activity/inactivity threshold resolution**

| WU_INACT_THS_W_[2:0] | 1 LSB resolution |
|---|---|
| 000 | 7.8125 m*g* |
| 001 | 15.625 m*g* |
| 010 | 31.25 m*g* |
| 111 | 62.5 m*g* |
| 100 | 125 m*g* |
| 101 | 250 m*g* |
| 110 | 250 m*g* |
| 111 | 250 m*g* |

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

When a certain number of consecutive X, Y, Z filtered data is smaller than the configured threshold, the OP_MODE_XL_[2:0] and the ODR_XL_[3:0] bits of the CTRL1 register are bypassed (inactivity) and the low-*g* accelerometer is internally set in low-power mode 1 at the sampling rate configured through the XL_INACT_ODR_[1:0] bits of the INACTIVITY_DUR register, although the content of the CTRL1 register is left untouched. The gyroscope behavior varies according to the configuration of the INACT_EN_[1:0] bits of the FUNCTIONS_ENABLE register. The duration of the inactivity state to be recognized is defined by the SLEEP_DUR_[3:0] bits of the WAKE_UP_DUR register: 1 LSB corresponds to 514 / ODR_XL time, where ODR_XL is the low-*g* accelerometer output data rate. If the SLEEP_DUR_[3:0] bits are set to 0000, the duration of the inactivity state to be recognized is equal to 18 / ODR_XL time.

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

When filtered data on one axis becomes bigger than the threshold for a configurable time, the CTRL1 register settings are immediately restored (activity) and the gyroscope is restored to the previous state. The duration of the activity state to be recognized is defined by the INACT_DUR_[1:0] bits of the INACTIVITY_DUR register. 1 LSB corresponds to 1 / ODR_XL time, where ODR_XL is the low-*g* accelerometer output data rate.

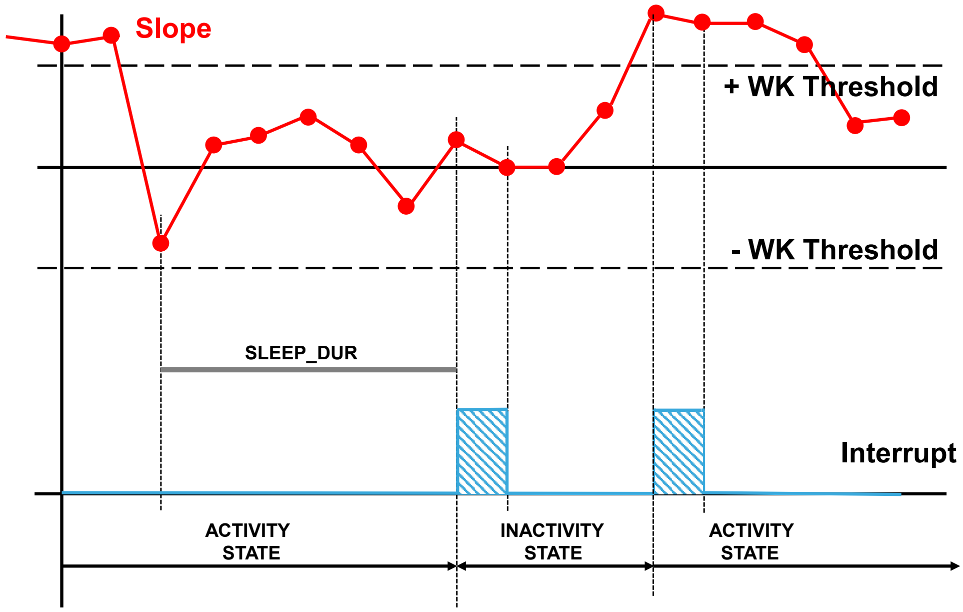
When the activity state is detected, the interrupt is set high for 1 / ODR_XL[s] period then it is automatically deasserted.

Once the activity/inactivity detection function is enabled, the activity/inactivity event can be driven to the two interrupt pins by setting the INT1_SLEEP_CHANGE bit of the MD1_CFG register or the INT2_SLEEP_CHANGE bit of the MD2_CFG register to 1. The activity/inactivity event can also be checked by reading the SLEEP_CHANGE_IA bit of the WAKE_UP_SRC or ALL_INT_SRC register.

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

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

**Figure 14. Activity/inactivity recognition (using the slope filter)**




A basic software routine for activity/inactivity detection is as follows:

| | | |
|---|---|---|
| 1. | Write 83h to FUNCTIONS_ENABLE | // Enable interrupt functions<br>// Set gyroscope to power-down mode when in inactivity state |
| 2. | Write 34h to INACTIVITY_DUR | // Set threshold resolution to 62.5 m*g*<br>// Set low-*g* accelerometer inactivity ODR to 15 Hz<br>// Set activity duration |
| 3. | Write 01h to INACTIVITY_THS | // Set threshold to 000001 |
| 4. | Write 05h to WAKE_UP_DUR | // Set the sleep duration to 0101 |
| 5. | Write 80h to MD1_CFG | // Activity/inactivity interrupt driven to INT1 pin |
| 6. | Write 04h to CTRL6 | // Set gyroscope FS = ±2000 dps |
| 7. | Write 02h to CTRL8 | // Set low-*g* accelerometer FS = ±8 *g* |
| 8. | Write 08h to CTRL1 | // Turn on the low-*g* accelerometer (ODR = 480 Hz) |
| 9. | Write 08h to CTRL2 | // Turn on the gyroscope (ODR = 480 Hz) |

In this example, the INACT_THS_[5:0] bits field of the INACTIVITY_THS register is set to 000001 and the resolution of 1 LSB is set to 62.5 m*g* (WU_INACT_THS_W_[2:0] bits of INACTIVITY_DUR register are set to 011), therefore the activity/inactivity threshold is 62.5 m*g*.

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 0101, corresponding to 5.35 s (= 5 * 514 / ODR_XL). After this period of time has elapsed, the low-*g* accelerometer ODR is internally set to 15 Hz (XL_INACT_ODR_[1:0] = 01) and the gyroscope is internally set to power-down mode.

The activity state is detected and the CTRL1 register settings are immediately restored and the gyroscope is turned on as soon as the slope data of (at least) one axis is bigger than the threshold for one sample, since the INACT_DUR_[1:0] bits of the INACTIVITY_DUR register are configured to 00.

### 5.7.1 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_EN_[1:0] bits of the FUNCTIONS_ENABLE register to 00.

## 5.8 High-*g* wake-up and shock detection

The high-*g* wake-up and shock-detection features can be used to generate an interrupt signal every time that a wake-up or shock event occurs on high-*g* accelerometer data, exploiting the wide full-scale range of the high-*g* accelerometer.

A wake-up event is detected every time that the acceleration signal along any of the X, Y, and Z axes exceeds a user-defined unsigned threshold. A shock is defined as an event that starts with a wake-up event and ends after a predetermined amount of time without wake-up events. The unsigned threshold value can be defined using the HG_WK_THS_[7:0] bits of the HG_WAKE_UP_THS register and the resolution of 1 LSB is 1 *g*. Moreover, it is possible to remove the high-*g* accelerometer sensor offset by setting a user-defined offset value on any axis. This can be done by setting the HG_USR_OFF_ON_WU bit of the CTRL1_XL_HG register to 1. Using this configuration, the user-defined offset is added to the input data for the high-*g* wake-up and shock detection functions. The programmable offset can be configured through the XL_HG_X_OFS_USR, XL_HG_Y_OFS_USR, and XL_HG_Z_OFS_USR registers (refer to Section 4.7: High-*g* accelerometer offset registers for more details).

A shock event ends when the acceleration signal along all the axes stays under the threshold defined by the HG_WK_THS_[7:0] bits for a configurable amount of time (shock duration). The high-*g* shock duration value to exit from the shock state can be defined using the HG_SHOCK_DUR_[3:0] bits of the HG_FUNCTIONS_ENABLE register and the equivalent duration in seconds can be computed as HG_SHOCK_DUR [s] = (HG_SHOCK_DUR[3:0] + 1) * 512 / ODR_XL_HG.

*Note: Allowed values for HG_SHOCK_DUR_[3:0] are in the range of 0 (default) to 14.*

The device is in the shock state for the entire time between the event and the exit from the shock event. This condition can be monitored through the HG_SHOCK_STATE bit in the HG_WAKE_UP_SRC register. If another wake-up event occurs while the device is waiting for HG_SHOCK_DUR to exit from the shock state, the HG_SHOCK_DUR timer is restarted.

The high-*g* wake-up and shock detection interrupt generation can be enabled by setting the HG_INTERRUPTS_ENABLE bit of the HG_FUNCTIONS_ENABLE register. The high-*g* wake-up interrupt signal can be routed to the INT1 / INT2 pin by setting the INT1_HG_WU / INT2_HG_WU bit in the HG_FUNCTIONS_ENABLE register to 1. The shock detection interrupt signals can be routed to the INT1 / INT2 pin by setting the INT1_HG_SHOCK_CHANGE / INT2_HG_SHOCK_CHANGE bit in the INACTIVITY_THS register. The HG_IA bit in the ALL_INT_SRC register is the "OR" combination of these two signals and can be used to check if an interrupt is related to the high-*g* wake-up or shock detection events.

The high-*g* wake-up and shock detection status can be determined by reading the status bits in the HG_WAKE_UP_SRC register. The HG_SHOCK_CHANGE_IA and HG_SHOCK_STATE bits can be used to determine the high-*g* shock status:

- HG_SHOCK_CHANGE_IA = 0 and HG_SHOCK_STATE = 0: the device is not in a shock state
- HG_SHOCK_CHANGE_IA = 1 and HG_SHOCK_STATE = 1: the device, previously not in a shock state, enters in a shock state
- HG_SHOCK_CHANGE_IA = 0 and HG_SHOCK_STATE = 1: the device is in a shock state and is waiting for the signal along all axes to stay under the HG_WK_THS for the HG_SHOCK_DUR time to exit from the shock state
- HG_SHOCK_CHANGE_IA = 1 and HG_SHOCK_STATE = 0: the device is exiting from the shock state

Similarly, the HG_WU_CHANGE_IA and HG_WU_IA bits can be used to determine the high-*g* wake-up status:

- HG_WU_CHANGE_IA = 0 and HG_WU_IA = 0: the device is not in a wake-up state
- HG_WU_CHANGE_IA = 1 and HG_WU_IA = 1: the device, previously not in wake-up, enters in a wake-up state
- HG_WU_CHANGE_IA = 0 and HG_WU_IA = 1: the device is waiting for the signal along all axes to go under the HG_WK_THS to exit from the wake-up state
- HG_WU_CHANGE_IA = 1 and HG_WU_IA = 0: the device is exiting from the wake-up state

The HG_X_WU, HG_Y_WU, and HG_Z_WU bits can be used to check which axis has triggered a shock event.

By configuring the HG_WU_CHANGE_INT_SEL bit in the HG_FUNCTIONS_ENABLE register, it is possible to select a different behavior for the high-*g* wake-up detection. If the HG_WU_CHANGE_INT_SEL is set to 0, an interrupt is generated each time that the acceleration signal along any of the X, Y, and Z axes is over the HG_WK_THS threshold. If the HG_WU_CHANGE_INT_SEL bit is set to 1, an interrupt is generated each time that the acceleration signal of the high-*g* accelerometer passes from all axes under the HG_WK_THS threshold to at least one axis over the HG_WK_THS threshold, or vice versa.

Since the HG_IA bit in the ALL_INT_SRC register contains the information coming from both the high-*g* wake-up and shock detection functions, if latched mode is intended to be used (LIR bit of TAP_CFG0 register set to 1), two different configurations and respective flows can be used in order to understand which of the two functions generated the interrupt.

**Flow #1:**

- If using DIS_RST_LIR_ALL_INT = 1, it is not needed to route both the interrupt functions to either INT1 or INT2 pin: either the wake-up or the shock can be routed to any interrupt pin.
- In order to clear the interrupt signal, it is necessary to read the HG_WAKE_UP_SRC register and the source registers of all the interrupt signals routed to the INT1 or INT2 pin.

**Flow #2:**

- If using DIS_RST_LIR_ALL_INT = 0, both the wake-up and shock detection functions must be routed to the INT1 or INT2 pin.
- It is necessary to read the ALL_INT_SRC register first and, if HG_IA = 1, the HG_WAKE_UP_SRC register after. Two cases are possible, based on the selected HG_WU_CHANGE_INT_SEL bit configuration:
- If HG_WU_CHANGE_INT_SEL = 0:
  - If HG_SHOCK_STATE = 1, the interrupt event is a wake-up or shock event.
  - If HG_SHOCK_STATE = 0, the interrupt event is a shock end event

*Note: Since it is required to read the ALL_INT_SRC before the HG_WAKE_UP_SRC register, the HG_WU_IA, HG_X_WU, HG_Y_WU, and HG_Z_WU bits in the HG_WAKE_UP_SRC register are always equal to zero.*

*Note: The HG_WU_CHANGE_IA bit must be ignored.*

- If HG_WU_CHANGE_INT_SEL = 1:
  - If HG_SHOCK_STATE = 1 and HG_WU_IA = 1, the interrupt event is a wake-up or shock event.
  - If HG_SHOCK_STATE = 1 and HG_WU_IA = 0, the interrupt event is an exit from the wake-up event.
  - If HG_SHOCK_STATE = 0, the interrupt event is a shock end event.

*Note: HG_WU_IA, HG_X_WU, HG_Y_WU, and HG_Z_WU bits in HG_WAKE_UP_SRC register are not latched and they are related to the last sample that has been generated.*

A basic software routine for the high-*g* wake-up event recognition is given below.

```
1. Write 32h to HG_WAKE_UP_THS         // Set high-g wake-up threshold to 50 g
2. Write 90h to HG_FUNCTIONS_ENABLE    // Enable high-g interrupt functions
                                       // High-g wake-up interrupt driven to the INT1 pin
3. Write 19h to CTRL1_XL_HG            // Turn on the high-g accelerometer (ODR = 480 Hz FS = ±64 g)
```

A basic software routine for the high-*g* shock event recognition is given below.

```
1. Write 32h to HG_WAKE_UP_THS         // Set high-g wake-up threshold to 50 g
2. Write 81h to HG_FUNCTIONS_ENABLE    // Enable high-g interrupt functions
                                       // Set shock duration to 1024 samples (that is about 2.13 s at 480 Hz ODR)
3. Write 40h to INACTIVITY_THS         // High-g shock interrupt driven to INT1 pin
4. Write 19h to CTRL1_XL_HG            // Turn on the high-g accelerometer (ODR = 480 Hz, FS = ±64 g)
```

## 5.9 Boot status

After the device is powered up, it performs a 10 ms (maximum) boot procedure to load the trimming parameters. After the boot is completed, the low-*g* accelerometer, the high-*g* accelerometer, and the gyroscope are automatically configured in power-down mode. During the boot time, the registers are not accessible.

*Note: If it is required to force a boot by removing and resupplying the VDD and the time between removing and resupplying the VDD is less than 20 ms, then the maximum boot time increases to 30 ms.*

After power-up, the trimming parameters can be reloaded by setting the BOOT bit of the CTRL3 register to 1. In this case, it is mandatory to wait 30 ms for the completion of the reboot internal procedure. The BOOT bit of the CTRL3 register automatically returns to 0.

If the reset to the default value of the control registers is required, it can be performed by setting the SW_RESET bit of the CTRL3 register to 1. When this bit is set to 1, the following registers are reset to their default value:

- FUNC_CFG_ACCESS (01h)
- ODR_TRIG_CFG (06h) through ALL_INT_SRC (1Dh)
- TIMESTAMP0 (40h) through TIMESTAMP3 (43h)
- WAKE_UP_SRC (45h) through D6D_SRC (47h)
- HG_WAKE_UP_SRC (4Ch) through CTRL1_XL_HG (4Eh)
- FUNCTIONS_ENABLE (50h) through UI_HANDSHAKE_CTRL (64h)
- FIFO_DATA_OUT_TAG (78h)

The software reset procedure takes a maximum of 150 µs. The status of the reset is signaled by the status of the SW_RESET bit of the CTRL3 register. Once the reset is completed, this bit is automatically set low.

The reboot flow is as follows:

1. Set the low-*g* accelerometer, high-*g* accelerometer, and gyroscope in power-down mode.
2. Set the BOOT bit of the CTRL3 register to 1.
3. Wait 30 ms.

The software reset flow is as follows:

1. Set the low-*g* accelerometer, high-*g* accelerometer, and gyroscope in power-down mode.
2. Set the SW_RESET bit of the CTRL3 register to 1.
3. Monitor the software reset status. There are two possibilities:
   a. Wait 150 µs.
   b. Poll the SW_RESET bit of the CTRL3 register until it returns to 0.

In order to avoid conflicts, the reboot and the software reset must not be executed at the same time (do not set to 1 at the same time both the BOOT bit and the SW_RESET bit of the CTRL3 register). The above flows must be performed serially.

If a complete reset (including the boot, software reset, and a reset of the embedded functions and internal filters) is required, it can be performed by setting the SW_POR bit of the FUNC_CFG_ACCESS register. When this bit is set to 1, the device triggers a complete reset of the device, analogous to a power-on-reset. In this case, it is mandatory to wait 30 ms for the completion of device reset. The SW_POR bit of the FUNC_CFG_ACCESS register automatically returns to 0. The complete reset flow is as follows:

1. Set the SW_POR bit of the FUNC_CFG_ACCESS register to 1.
2. Wait 30 ms.

# 6 Embedded functions

The device implements in the 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
- Sensor fusion functions (game rotation vector, gravity vector, gyroscope bias)

## 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 30 Hz and are based on the low-*g* accelerometer sensor only. Consequently, the low-*g* accelerometer ODR must be set at a value of 30 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 low-*g* 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 s of device inactivity.

An additional false-positive rejection (FPR) block can be enabled to perform the real-time recognition of the walking activity (including running) based on statistical data and to inhibit the step counter if no walking activity is detected. It can be activated as follows:

- Set the FP_REJECTION_EN bit of the PEDO_CMD_REG embedded advanced features register to 1.
- Set either the MLC_EN bit of the EMB_FUNC_EN_B or the MLC_BEFORE_FSM_EN bit of the EMB_FUNC_EN_A to 1.

In the LSM6DSV80X device, the FPR block can be customized by the user. In this case, the MLC must be programmed in order to use the first decision tree for the recognition of two classes: no walk (class with code 0x04) and walk (class with code 0x08). In detail, the step counter is inhibited if the following group of classes is detected by the MLC:

- Classes with code 0x4 through 0x7
- Classes with code 0xC through 0xE

STMicroelectronics provides the tools to generate specific pedometer configurations starting from a set of datalogs with a reference number of steps (MEMS Studio GUI on www.st.com).

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

**Table 48. 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 FUNCTIONS_ENABLE 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 5.6 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 to 0.
- STEP_OVERFLOW: overflow signal that 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 (page 1) registers, this bit is kept to 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 field (concatenation of PEDO_SC_DELTAT_H and PEDO_SC_DELTAT_L embedded advanced features registers) and the CARRY_COUNT_EN bit in the PEDO_CMD_REG embedded advanced features register.

**Table 49. 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 / MAX_RATE seconds, where MAX_RATE denotes the maximum rate of the enabled embedded functions. If only the pedometer function is enabled, the duration of the pulse is then equal to 1 / 30 seconds. 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 8: 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 80h to FUNC_CFG_ACCESS // Enable access to embedded functions registers
2. Write 40h to PAGE_RW // Select write operation mode
3. Write 11h to PAGE_SEL // Select page 1
4. Write 83h to PAGE_ADDR // Set embedded advanced features register to be written (PEDO_CMD_REG)
5. Write 04h to PAGE_VALUE // Enable false-positive rejection block link with pedometer (FP_REJECTION_EN = 1)
6. Write 00h to PAGE_RW // Write operation mode disabled
7. Write 08h to EMB_FUNC_EN_A // Enable pedometer
8. Write 10h to EMB_FUNC_EN_B // Enable false-positive rejection block (MLC_EN = 1)
9. Write 08h to EMB_FUNC_INT1 // Step detection interrupt driven to INT1 pin
10. Write 00h to FUNC_CFG_ACCESS // Disable access to embedded functions registers
11. Write 02h to MD1_CFG // Enable routing the embedded functions interrupt
12. Write 02h to CTRL8 // FS_XL = ±8 *g*
13. Write 04h to CTRL1 // Turn on the low-*g* accelerometer (ODR_XL = 30 Hz)

## 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 low-*g* 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 30 Hz, so the low-*g* accelerometer ODR must be set at a value of 30 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 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 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 / MAX_RATE seconds, where MAX_RATE denotes the maximum rate of the enabled embedded functions. If only the significant motion function is enabled, the duration of the pulse is then equal to 1 / 30 seconds. 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 that shows how to enable significant motion detection is as follows:

1. Write 80h to FUNC_CFG_ACCESS // Enable access to embedded functions registers
2. Write 20h to EMB_FUNC_EN_A // Enable significant motion detection
3. Write 20h to EMB_FUNC_INT1 // Significant motion interrupt driven to INT1 pin
4. Write 80h to PAGE_RW // Enable latched mode for embedded functions
5. Write 00h to FUNC_CFG_ACCESS // Disable access to embedded functions registers
6. Write 02h to MD1_CFG // Enable routing the embedded functions interrupt
7. Write 04h to CTRL1 // Turn on the low-*g* accelerometer
   // ODR_XL = 30 Hz

## 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 low-*g* accelerometer.

The tilt function works at 30 Hz, so the low-*g* accelerometer ODR must be set at a value of 30 Hz or higher.

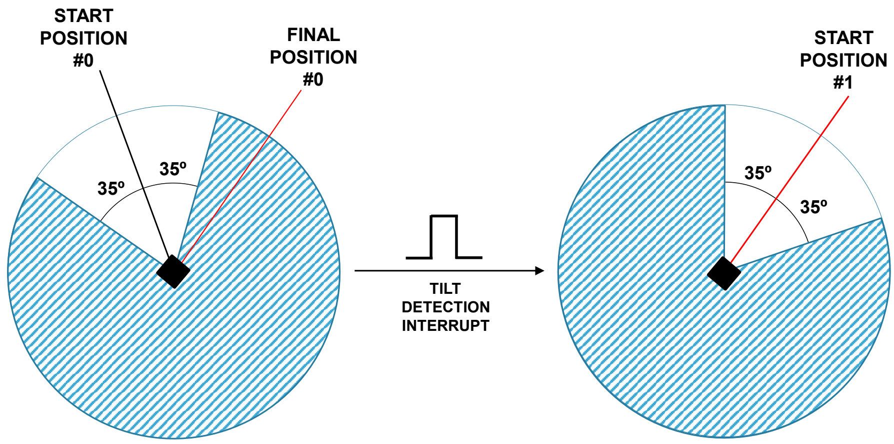
In order to enable the relative tilt detection function, it is necessary to set the TILT_EN bit of the EMB_FUNC_EN_A embedded functions register to 1. The algorithm can be reinitialized by asserting the TILT_INIT bit of the EMB_FUNC_INIT_A embedded functions register.

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/reinitialized 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 15. Tilt example 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 15. 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 routing the embedded functions event 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 / MAX_RATE seconds, where MAX_RATE denotes the maximum rate of the enabled embedded functions. If only the tilt function is enabled, the duration of the pulse is then equal to 1 / 30 seconds. 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 that shows how to enable the tilt detection function:

| | | |
|---|---|---|
| 1. | Write 80h to FUNC_CFG_ACCESS | // Enable access to embedded functions registers |
| 2. | Write 10h to EMB_FUNC_EN_A | // Enable tilt detection |
| 3. | Write 10h to EMB_FUNC_INT1 | // Tilt interrupt driven to INT1 pin |
| 4. | Write 80h to PAGE_RW | // Enable latched mode for embedded functions |
| 5. | Write 00h to FUNC_CFG_ACCESS | // Disable access to embedded functions registers |
| 6. | Write 02h to MD1_CFG | // Enable routing the embedded functions interrupt |
| 7. | Write 04h to CTRL1 | // Turn on the low-*g* accelerometer<br>// ODR_XL = 30 Hz |

## 6.4 Timestamp

Together with sensor data the device can provide timestamp information.

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

The nominal timestamp resolution is 21.7 µs. It is possible to get the actual timestamp resolution value through the FREQ_FINE_[7:0] bits of the INTERNAL_FREQ_FINE register, which contains the difference in percentage of the actual ODR (and timestamp rate) with respect to the nominal value.

$$t_{actual}[s] = \frac{1}{46080 \cdot (1 + 0.0013 \cdot FREQ\_FINE)}$$

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

$$ODR_{actual}[Hz] = \frac{7680 \cdot (1 + 0.0013 \cdot FREQ\_FINE)}{ODR_{coeff}}$$

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

**Table 50. $ODR_{coeff}$ values**

| Selected ODR [Hz] | ODRcoeff |
|---|---|
| 7.5 | 1024 |
| 15 | 512 |
| 30 | 256 |
| 60 | 128 |
| 120 | 64 |
| 240 | 32 |
| 480 | 16 |
| 960 | 8 |
| 1920 | 4 |
| 3840 | 2 |
| 7680 | 1 |

If all the sensors are 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 26 hours, the counter is automatically reset to 00000000h and continues to count. The timer count can be reset to zero at any time by writing the reset value AAh in the TIMESTAMP2 register. The reset procedure can be performed only if at least one sensor is in any active operating mode. After the reset value has been written to the register, at least 400 µs must elapse before performing any other write transaction.

The TIMESTAMP_ENDCOUNT bit of the ALL_INT_SRC goes high 5.6 ms before the occurrence of a timestamp overrun condition. This flag is reset when the ALL_INT_SRC register is read. It is also possible to route this signal to the INT2 pin (65 µs duration pulse) by setting the INT2_TIMESTAMP bit of MD2_CFG to 1.

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

## 6.5 Sensor fusion functions

A dedicated sensor fusion block SFLP (sensor fusion low power) is available for generating the following data based on the low-*g* accelerometer and gyroscope data processing:

- Game rotation vector, which provides a quaternion representing the attitude of the device. The game rotation vector is stored in the dedicated registers SFLP_QUATW_L/H, SFLP_QUATX_L/H, SFLP_QUATY_L/H, SFLP_QUATZ_L/H as 16-bit floating-point values (that is, formatted as SEEEEEFFFFFFFFFF where S = 1 sign bit, E = 5 exponent bits, F = 10 fraction bits)
- Gravity vector, which provides a three-dimensional vector representing the direction of gravity. The gravity vector is stored in the dedicated registers SFLP_GRAVX_L/H, SFLP_GRAVY_L/H, SFLP_GRAVZ_L/H as 16-bit words in two's complement with 0.0061 m*g*/LSB sensitivity
- Gyroscope bias, which provides a three-dimensional vector representing the gyroscope bias. The gravity vector is stored in the dedicated registers SFLP_GBIASX_L/H, SFLP_GBIASY_L/H, SFLP_GBIASZ_L/H as 16-bit words in two's complement with 4.375 mdps/LSB sensitivity

The SFLP block is enabled by setting the SFLP_GAME_EN bit to 1 of the EMB_FUNC_EN_A embedded functions register.

The SFLP block can be reinitialized by setting the SFLP_GAME_INIT bit to 1 of the EMB_FUNC_INIT_A embedded functions register.

The SFLP block works at a configurable output data rate (which must be equal to or less than the selected output data rates of the low-*g* accelerometer and gyroscope) through the SFLP_GAME_ODR_[2:0] field of the SFLP_ODR embedded functions register according to the following values:

- 000: 15 Hz
- 001: 30 Hz
- 010: 60 Hz
- 011: 120 Hz (default)
- 100: 240 Hz
- 101: 480 Hz

SFLP-generated data can be read from the dedicated registers described above or from the FIFO if enabled, see Section 8: First-in, first-out (FIFO) buffer.

The typical supply current of the SFLP block is indicated in Table 51.

**Table 51. SFLP supply current (@VDD = 1.8 V, T = 25°C)**

| SFLP ODR [Hz] | Supply current [µA] |
| --- | --- |
| 15 | 3.5 |
| 30 | 7 |
| 60 | 14 |
| 120 | 28 |
| 240 | 56 |
| 480 | 112 |

### 6.5.1 Gyroscope bias initial value setting

The SFLP embeds a gyroscope bias calibration routine, which is automatically executed when the device is steady. In applications where a steady condition for the gyroscope bias calibration cannot be guaranteed, a specific flow is needed to set a previously computed bias in the SFLP block. This procedure forces a reset of the SFLP algorithm and must be implemented as follows:

1. Convert gbias in HFP format in [rad/s] and divide by the k factor according to Table 52.
2. Write 80h to the FUNC_CFG_ACCESS register to enable access to the embedded functions registers.
3. If the SFLP block is on, turn it off by setting the SFLP_GAME_EN bit to 0.
4. Write the gbias values computed at step 1 to registers SFLP_GBIASX_INIT_L/H, SFLP_GBIASY_INIT_L/H, SFLP_GBIASZ_INIT_L/H (from address 32h to 38h).
5. Turn the SFLP block on by setting the SFLP_GAME_EN bit to 1.
6. Write 00h to the FUNC_CFG_ACCESS register to disable access to the embedded functions registers.

**Table 52. k factor**

| SFLP game ODR [Hz] | k factor |
|---|---|
| 15 | 0.04 |
| 30 | 0.02 |
| 60 | 0.01 |
| 120 | 0.005 |
| 240 | 0.0025 |
| 480 | 0.00125 |

## 6.6 Embedded functions additional configurations and monitoring

The device provides the possibility to enable some additional configurations if needed through the EMB_FUNC_CFG register.

It allows three additional features:

- EMB_FUNC_IRQ_MASK_XL_SETTL bit can be set to 1 to enable masking the execution trigger of the embedded functions when the low-*g* accelerometer data are in the settling phase, in order to avoid processing the low-*g* accelerometer data during the settling phase.
- EMB_FUNC_IRQ_MASK_XL_HG_SETTL bit can be set to 1 to enable masking the execution trigger of the embedded functions when the high-*g* accelerometer data are in the settling phase, in order to avoid processing the high-*g* accelerometer data during the settling phase.
- EMB_FUNC_IRQ_MASK_G_SETTL bit can be set to 1 to enable masking the execution trigger of the embedded functions when gyroscope data are in the settling phase, in order to avoid processing the gyroscope data during the settling phase.
- EMB_FUNC_DISABLE bit can be set to 1 to stop the execution trigger of the embedded functions. When this bit is set back to 0, all the initialization procedures are forced and the execution trigger is again enabled.

The device provides the EMB_FUNC_SENSOR_CONV_EN register to allow the user to selectively enable/ disable the data conversion internal block for each of the sensors available as embedded functions input except for the low-*g* accelerometer, for which the conversion is always applied. This feature is intended to reduce the overhead in terms of supply current and execution time for the configurations that do not use all the sensors as input and it allows skipping data conversion for the unused sensors. The data conversion in the embedded functions is enabled by default for all the sensors, but the user can set to 0 one or more dedicated bits in EMB_FUNC_SENSOR_CONV_EN to disable the conversion for the unused sensors. The bits that can be used to enable / disable data conversion for each sensor (except for the low-*g* accelerometer) are listed below as well as their description:

- EXT_SENSOR_CONV_EN: enables / disables the external sensor data conversion for the embedded functions.
- TEMP_CONV_EN: enables / disables the temperature data conversion for the embedded functions.
- GYRO_CONV_EN: enables / disables the gyroscope data conversion for the embedded functions.
- XL_HG_CONV_EN: enables / disables the high-*g* accelerometer data conversion for the embedded functions.

The device provides the capability to monitor the execution of the embedded functions through the EMB_FUNC_EXEC_STATUS embedded functions register.

It contains the following information:

- Execution time overrun: this information is contained in the EMB_FUNC_EXEC_OVR bit. It is asserted if the execution time of the enabled embedded functions exceeds the maximum time, that is, a new set of sensor data to be used as input is generated before the end of the embedded functions execution.
- Execution ongoing: this information is contained in the EMB_FUNC_ENDOP bit. When this bit is set to 1, no embedded function is running, while when this bit is set to 0, embedded functions are running. This information can be routed to the INT2 pin by setting the INT2_EMB_FUNC_ENDOP bit of the INT2_CTRL register.

# 7 Mode 2 - sensor hub mode

The hardware flexibility of the LSM6DSV80X allows connecting the pins with different mode connections to external sensors to expand functionalities such as adding a sensor hub. When sensor hub mode (mode 2) is enabled, both the I²C/MIPI I3C®/SPI (3- and 4-wire) target interface and the I²C controller interface for the connection of external sensors are available. Mode 2 connection mode is described in detail in the following paragraphs.

## 7.1 Sensor hub mode description

In sensor hub mode (mode 2) up to four external sensors can be connected to the I²C controller interface of the device. The sensor hub trigger signal can be synchronized with the low-*g* accelerometer/gyroscope data-ready signal (up to 480 Hz). In this configuration, the sensor hub ODR can be configured through the SHUB_ODR_[2:0] bits of the TGT0_CONFIG register. Alternatively, an external signal connected to the INT2 pin can be used as the sensor hub trigger. In this second case, the maximum ODR supported for external sensors depends on the number of read / write operations that can be executed between two consecutive trigger signals.

On the sensor hub trigger signal, all the write and read I²C operations configured through the registers TGTx_ADD, TGTx_SUBADD, TGTx_CONFIG, and DATAWRITE_TGT0 are performed sequentially from external sensor 0 to external sensor 3 (depending on the external sensors enabled through the AUX_SENS_ON[1:0] field in the CONTROLLER_CONFIG register).

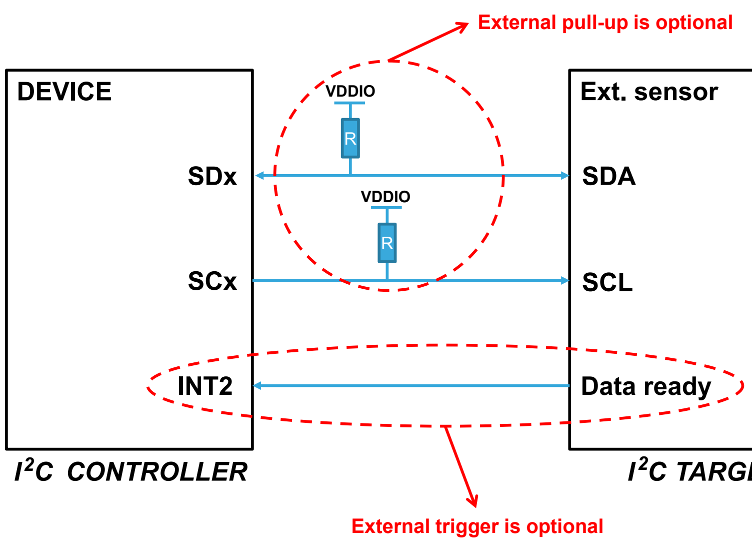
External sensor data can also be stored in FIFO (see Section 8: First-in, first-out (FIFO) buffer for details).

If both the low-*g* accelerometer and the gyroscope are in power-down mode, the sensor hub does not work.

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

The SHUB_PU_EN bit of the IF_CFG register can be used to enable or disable the internal pull-up on the I²C controller line. When this bit is set to 0, the internal pull-up is disabled and the external pull-up resistors on the SDx/SCx pins are required, as shown in Figure 16. External sensor connections in mode 2. When this bit is set to 1, the internal pull-up is enabled (regardless of the configuration of the CONTROLLER_ON bit) and the external pull-up resistors on the SDx/SCx pins are not required.

**Figure 16. External sensor connections in mode 2**



## 7.2 Sensor hub mode registers

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

The CONTROLLER_CONFIG register has to be used for the configuration of the I²C controller interface.

A set of registers TGTx_ADD, TGTx_SUBADD, TGTx_CONFIG is dedicated to the configuration of the four target interfaces associated to the four connectable external sensors. An additional register, DATAWRITE_TGT0, is associated with target #0 only. It has to be used to implement the write operations.

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

### 7.2.1 CONTROLLER_CONFIG (14h)

This register is used to configure the I²C controller interface.

Table 53. **CONTROLLER_CONFIG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| RST_CONTROLLER_REGS | WRITE_ONCE | START_CONFIG | PASS_THROUGH_MODE | 0 | CONTROLLER_ON | AUX_SENS_ON1 | AUX_SENS_ON0 |

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

*Note: The WRITE_ONCE bit must be set to 1 if target 0 is used for read transactions.*

- START_CONFIG bit selects the sensor hub trigger signal.
  - When this bit is set to 0, the low-*g* accelerometer/gyroscope sensor has to be active (not in power-down mode) and the sensor hub trigger signal is the low-*g* accelerometer/gyroscope data-ready signal, with a frequency defined by the SHUB_ODR_[2:0] bits of the TGT0_CONFIG register (up to 480 Hz).
  - When this bit is set to 1, at least one sensor between the low-*g* accelerometer and the gyroscope has to be active and the sensor hub trigger signal is the INT2 pin. In fact, when both the CONTROLLER_ON bit and START_CONFIG bit are set to 1, the INT2 pin is configured as an input signal. In this case, the INT2 pin has to be connected to the data-ready pin of the external sensor (Figure 16. External sensor connections in mode 2) in order to trigger the read/write operations on the external sensor registers. The sensor hub interrupt from INT2 polarity can be selected through the INT2_IN_LH bit of the CTRL4 register: if it is set to 0, the pin is active low, otherwise, it is active high.

*Note: In case of external trigger signal usage (START_CONFIG=1), if the INT2 pin is connected to the data-ready pin of the external sensor (Figure 16. External sensor connections in mode 2) and the latter is in power-down mode, then no data-ready signal can be generated by the external sensor. For this reason, the initial configuration of the external sensor's register has to be performed using the internal trigger signal (START_CONFIG=0). After the external sensor is activated and the data-ready signal is available, the external trigger signal can be used by switching the START_CONFIG bit to 1.*

- PASS_THROUGH_MODE bit is used to enable/disable the I²C interface pass-through. When this bit is set to 1, the main I²C line (for example, connected to an external microcontroller) is short-circuited with the auxiliary one, in order to implement a direct access to the external sensor registers. See Section 7.3: Sensor hub pass-through feature for details.

- CONTROLLER_ON bit has to be set to 1 to enable the auxiliary I²C controller of the device (sensor hub mode). In order to change the sensor hub configuration at runtime or when setting the low-*g* accelerometer and gyroscope sensor in power-down mode, or when applying the software reset procedure, the I²C controller must be disabled, followed by a 300 µs wait. The following procedure must be implemented:
  1. Turn off the I²C controller by setting CONTROLLER_ON = 0.
  2. Wait 300 µs.
  3. Change the configuration of the sensor hub registers or set the low-*g* accelerometer/gyroscope in power-down mode or apply the software reset procedure.
- AUX_SENS_ON[1:0] bits have to be set accordingly to the number of targets to be used. I²C transactions are performed sequentially from target 0 to target 3. The possible values are:
  – 00: one target
  – 01: two targets
  – 10: three targets
  – 11: four targets

### 7.2.2 STATUS_CONTROLLER (22h)

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

**Table 54. STATUS_CONTROLLER / STATUS_CONTROLLER_MAINPAGE register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| WR_ONCE_DONE | TARGET3_NACK | TARGET2_NACK | TARGET1_NACK | TARGET0_NACK | 0 | 0 | SENS_HUB_ENDOP |

- WR_ONCE_DONE bit is set to 1 after a write operation performed with the WRITE_ONCE bit configured to 1 in the CONTROLLER_CONFIG register. This bit can be polled in order to check if the single write transaction has been completed.
- TARGETx_NACK bits are set to 1 if a “not acknowledge” event happens during the communication with the corresponding target x.
- SENS_HUB_ENDOP bit reports the end of an I²C controller transaction. It is set to 1 when the transaction is concluded; it is reset to 0 when the STATUS_CONTROLLER / STATUS_CONTROLLER_MAINPAGE register is read. When a sensor hub routine is completed, this bit automatically goes to 1 and the external sensor data are available to be read from the SENSOR_HUB_x registers (depending on the configuration of the TGTx_ADD, TGTx_SUBADD, TGTx_CONFIG registers). Information about the status of the I²C controller can be driven to the INT1 interrupt pin by setting the INT1_SHUB bit of the MD1_CFG register to 1. This signal goes high on a rising edge of the SENS_HUB_ENDOP signal and it is cleared only if the STATUS_CONTROLLER / STATUS_CONTROLLER_MAINPAGE register is read.

### 7.2.3 TGT0_ADD (15h), TGT0_SUBADD (16h), TGT0_CONFIG (17h)

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

**Table 55. TGT0_ADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target0_add6 | target0_add5 | target0_add4 | target0_add3 | target0_add2 | target0_add1 | target0_add0 | rw_0 |

- The target0_add[6:0] bits are used to indicate the I²C target address of the first external sensor.
- The rw_0 bit configures the read/write operation to be performed on the first external sensor (0: write operation; 1: read operation). The read/write operation is executed when the next sensor hub trigger event occurs.

**Table 56. TGT0_SUBADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target0_reg7 | target0_reg6 | target0_reg5 | target0_reg4 | target0_reg3 | target0_reg2 | target0_reg1 | target0_reg0 |

- The target0_reg[7:0] bits are used to indicate the address of the register of the first external sensor to be written (if the rw_0 bit of the TGT0_ADD register is set to 0) or the address of the first register to be read (if the rw_0 bit is set to 1).

**Table 57. TGT0_CONFIG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| SHUB_ ODR_2 | SHUB_ ODR_1 | SHUB_ ODR_0 | 0 | BATCH_EXT_ SENS_0_EN | target0 _numop2 | target0 _numop1 | target0 _numop0 |

- The SHUB_ODR_[2:0] bits are used to configure the sensor hub output data rate when using an internal trigger (low-*g* accelerometer/gyroscope data-ready signals). The sensor hub output data rate can be configured to six possible values, limited by the ODR of the low-*g* accelerometer and gyroscope sensors:
  - 000: 1.875 Hz
  - 001: 15 Hz
  - 010: 30 Hz
  - 011: 60 Hz
  - 100: 120 Hz (default)
  - 101: 240 Hz
  - 110: 480 Hz

The maximum allowed value for the SHUB_ODR_[2:0] bits corresponds to the maximum ODR between the low-*g* accelerometer and gyroscope sensors. If the gyroscope is configured in sleep mode, its ODR must be considered equal to 0 Hz regardless of the configuration of the ODR_G_[3:0] bits.

- The BATCH_EXT_SENS_0_EN bit is used to enable batching the external sensor data associated to target0 in FIFO.
- The target0_numop[2:0] bits define the number of consecutive read operations to be performed on the first external sensor starting from the register address indicated in the TGT0_SUBADD register.

### 7.2.4 TGT1_ADD (18h), TGT1_SUBADD (19h), TGT1_CONFIG (1Ah)

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

**Table 58. TGT1_ADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target1_add6 | target1_add5 | target1_add4 | target1_add3 | target1_add2 | target1_add1 | target1_add0 | r_1 |

- The target1_add[6:0] bits are used to indicate the I²C target address of the second external sensor.
- The r_1 bit enables/disables the read operation to be performed on the second external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

**Table 59. TGT1_SUBADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target1_reg7 | target1_reg6 | target1_reg5 | target1_reg4 | target1_reg3 | target1_reg2 | target1_reg1 | target1_reg0 |

- The target1_reg[7:0] bits are used to indicate the address of the register of the second external sensor to be read when the r_1 bit of the TGT1_ADD register is set to 1.

**Table 60. TGT1_CONFIG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | BATCH_EXT_ SENS_1_EN | target1 _numop2 | target1 _numop1 | target1 _numop0 |

- The BATCH_EXT_SENS_1_EN bit is used to enable batching the external sensor data associated to target1 in FIFO.
- The target1_numop[2:0] bits define the number of consecutive read operations to be performed on the second external sensor starting from the register address indicated in the TGT1_SUBADD register.

### 7.2.5 TGT2_ADD (1Bh), TGT2_SUBADD (1Ch), TGT2_CONFIG (1Dh)

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

**Table 61. TGT2_ADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target2_add6 | target2_add5 | target2_add4 | target2_add3 | target2_add2 | target2_add1 | target2_add0 | r_2 |

- The target2_add[6:0] bits are used to indicate the I²C target address of the third external sensor.
- The r_2 bit enables/disables the read operation to be performed on the third external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

**Table 62. TGT2_SUBADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target2_reg7 | target2_reg6 | target2_reg5 | target2_reg4 | target2_reg3 | target2_reg2 | target2_reg1 | target2_reg0 |

- The target2_reg[7:0] bits are used to indicate the address of the register of the third external sensor to be read when the r_2 bit of the TGT2_ADD register is set to 1.

**Table 63. TGT2_CONFIG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | BATCH_EXT_ SENS_2_EN | target2 _numop2 | target2 _numop1 | target2 _numop0 |

- The BATCH_EXT_SENS_2_EN bit is used to enable batching the external sensor data associated to target2 in FIFO.
- The target2_numop[2:0] bits define the number of consecutive read operations to be performed on the third external sensor starting from the register address indicated in the TGT2_SUBADD register.

### 7.2.6 TGT3_ADD (1Eh), TGT3_SUBADD (1Fh), TGT3_CONFIG (20h)

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

**Table 64. TGT3_ADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target3_add6 | target3_add5 | target3_add4 | target3_add3 | target3_add2 | target3_add1 | target3_add0 | r_3 |

- The target3_add[6:0] bits are used to indicate the I²C target address of the fourth external sensor.
- The r_3 bit enables/disables the read operation to be performed on the fourth external sensor (0: read operation disabled; 1: read operation enabled). The read operation is executed when the next sensor hub trigger event occurs.

**Table 65. TGT3_SUBADD register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target3_reg7 | target3_reg6 | target3_reg5 | target3_reg4 | target3_reg3 | target3_reg2 | target3_reg1 | target3_reg0 |

- The target3_reg[7:0] bits are used to indicate the address of the register of the fourth external sensor to be read when the r_3 bit of the TGT3_ADD register is set to 1.

**Table 66. TGT3_CONFIG register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | BATCH_EXT_ SENS_3_EN | target3 _numop2 | target3 _numop1 | target3 _numop0 |

- The BATCH_EXT_SENS_3_EN bit is used to enable batching the external sensor data associated to target3 in FIFO.
- The target3_numop[2:0] bits define the number of consecutive read operations to be performed on the fourth external sensor starting from the register address indicated in the TGT3_SUBADD register.

### 7.2.7 DATAWRITE_TGT0 (21h)

**Table 67. DATAWRITE_TGT0 register**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| target0_ dataw7 | target0_ dataw6 | target0_ dataw5 | target0_ dataw4 | target0_ dataw3 | target0_ dataw2 | target0_ dataw1 | Target0_ dataw0 |

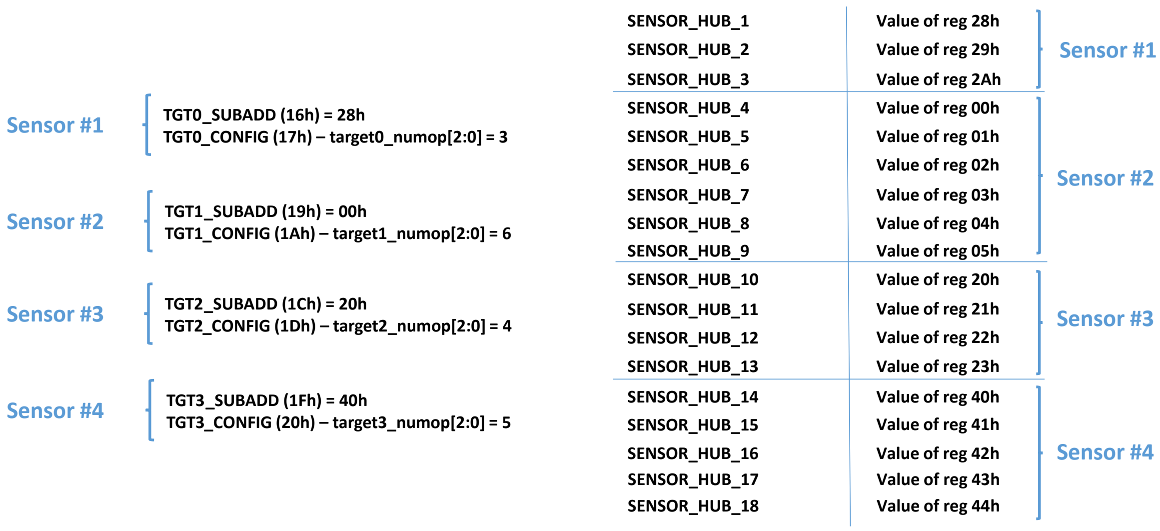
- The target0_dataw[7:0] bits are dedicated, when the rw_0 bit of the TGT0_ADD register is set to 0 (write operation), to indicate the data to be written to the first external sensor at the address specified in the TGT0_SUBADD register.

### 7.2.8 SENSOR_HUB_x registers

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

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

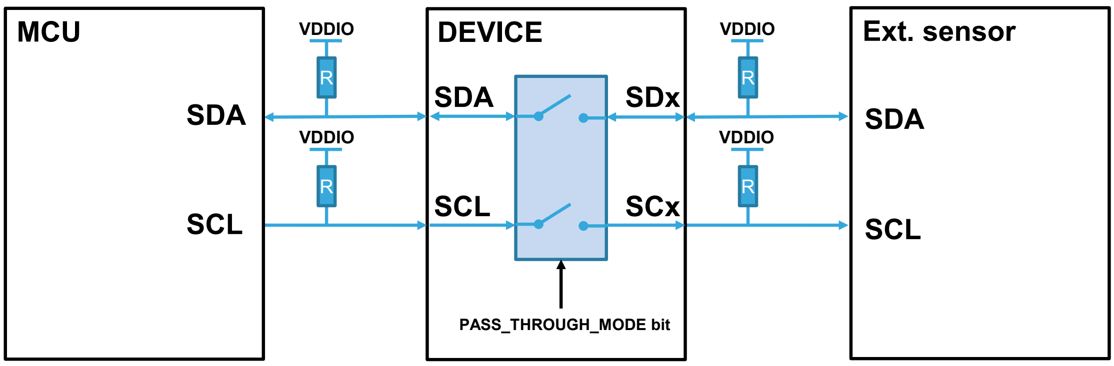
Figure 17. SENSOR_HUB_X allocation example



## 7.3 Sensor hub pass-through feature

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

Figure 18. **Pass-through feature**




The following procedure can be implemented to enable the pass-through mode:

1. If the I²C controller is enabled (CONTROLLER_ON = 1), turn it off (set the CONTROLLER_ON bit to 0) and wait 300 µs.
2. If the pull-up on the I²C controller line is enabled, disable it (set the SHUB_PU_EN bit of the IF_CFG register to 0).
3. Enable the pass-through mode by setting the PASS_THROUGH_MODE bit to 1.

## 7.4 Sensor hub mode example

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

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

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

The **one-shot read routine** (using internal trigger) is described below. For simplicity, the routine uses the low-*g* accelerometer configured at 120 Hz, with external pull-ups on the I²C auxiliary bus.

```
1.  Write 40h to FUNC_CFG_ACCESS           // Enable access to sensor hub registers
2.  Write EXT_SENS_ADDR | 01h to TGT0_ADD  // Configure external device address (EXT_SENS_ADDR)
                                           // Enable read operation (rw_0 = 1)
3.  Write REG to TGT0_SUBADD               // Configure address (REG) of the register to be read
4.  Write 81h to TGT0_CONFIG               // Read one byte, SHUB_ODR = 120 Hz
5.  Write 44h to CONTROLLER_CONFIG         // WRITE_ONCE is mandatory for read
                                           // I²C controller enabled, using target 0 only
6.  Write 00h to FUNC_CFG_ACCESS           // Disable access to sensor hub registers
7.  Read OUTX_H_A register                 // Clear low-g accelerometer data-ready XLDA
8.  Poll STATUS_REG, until XLDA = 1        // Wait for sensor hub trigger
```

| | | |
|---|---|---|
| 9. | Poll STATUS_CONTROLLER_MAINPAGE, until SENS_HUB_ENDOP = 1 | // Wait for sensor hub read transaction |
| 10. | Write 40h to FUNC_CFG_ACCESS | // Enable access to sensor hub registers |
| 11. | Write 00h to CONTROLLER_CONFIG | // I²C controller disable |
| 12. | Wait 300 µs | |
| 13. | Read SENSOR_HUB_1 register | // Retrieve the output of the read operation |
| 14. | Write 00h to FUNC_CFG_ACCESS | // Disable access to sensor hub registers |

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

| | | |
|---|---|---|
| 1. | Write 40h to FUNC_CFG_ACCESS | // Enable access to sensor hub registers |
| 2. | Write EXT_SENS_ADDR \| 01h to TGT0_ADD | // Configure external device address (EXT_SENS_ADDR)<br>// Enable read operation (rw_0 = 1) |
| 3. | Write REG to TGT0_SUBADD | // Configure address (REG) of the register to be read |
| 4. | Write 8xh to TGT0_CONFIG | // Read x bytes (up to six), SHUB_ODR = 120 Hz |
| 5. | Write 44h to CONTROLLER_CONFIG | // WRITE_ONCE is mandatory for read<br>// I²C controller enabled, using target 0 only |
| 6. | Write 00h to FUNC_CFG_ACCESS | // Disable access to sensor hub registers |

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

The **One-shot write routine** (using internal trigger) is described below. For simplicity, the routine uses the low-*g* accelerometer configured at 120 Hz, with external pull-ups on the I²C auxiliary bus.

| | | |
|---|---|---|
| 1. | Write 40h to FUNC_CFG_ACCESS | // Enable access to sensor hub registers |
| 2. | Write EXT_SENS_ADDR to TGT0_ADD | // Configure external device address (EXT_SENS_ADDR)<br>// Enable write operation (rw_0 = 0) |
| 3. | Write REG to TGT0_SUBADD | // Configure address (REG) of the register to be written |
| 4. | Write 80h to TGT0_CONFIG | // SHUB_ODR = 120 Hz |
| 5. | Write VAL to DATAWRITE_TGT0 | // Configure value (VAL) to be written in REG |
| 6. | Write 44h to CONTROLLER_CONFIG | // WRITE_ONCE enabled for single write<br>// I²C controller enabled, using target 0 only |
| 7. | Poll STATUS_CONTROLLER, until WR_ONCE_DONE = 1 | // Wait for sensor hub write transaction |
| 8. | Write 00h to CONTROLLER_CONFIG | // I²C controller disabled |
| 9. | Wait 300 µs | |
| 10. | Write 00h to FUNC_CFG_ACCESS | // Disable access to sensor hub registers |

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

| | | |
|---|---|---|
| 1. | Write 06h to CTRL1 | // Turn on the low-*g* accelerometer (for trigger signal) at 120 Hz |
| 2. | Perform **one-shot read** with | // Check LIS2MDL WHO_AM_I register |
| | TGT0_ADD = 3Dh | // LIS2MDL target address is 3Ch and rw_0=1 |
| | TGT0_SUBADD = 4Fh | // WHO_AM_I register address is 4Fh |
| 3. | Perform **one-shot write** with | // Write LIS2MDL register CFG_REG_A (60h) = 8Ch |
| | TGT0_ADD = 3Ch | // LIS2MDL target address is 3Ch and rw_0=0 |
| | TGT0_SUBADD = 60h | // Enable temperature compensation |
| | DATAWRITE_TGT0 = 8Ch | // Enable magnetometer at 100 Hz ODR in continuous mode |
| 4. | Perform **one-shot write** with | // Write LIS2MDL register CFG_REG_B (61h) = 02h |
| | TGT0_ADD = 3Ch | // LIS2MDL target address is 3Ch and rw_0=0 |
| | TGT0_SUBADD = 61h | // Enable magnetometer offset cancellation |
| | DATAWRITE_TGT0 = 02h | |
| 5. | Perform **one-shot write** with | // Write LIS2MDL register CFG_REG_B (62h) = 10h |
| | TGT0_ADD = 3Ch | // LIS2MDL target address is 3Ch and rw_0 = 0 |
| | TGT0_SUBADD = 62h | // Enable magnetometer BDU |
| | DATAWRITE_TGT0 = 10h | |
| 6. | Set up **continuous read** with | // LIS2MDL target address is 3Ch and rw_0 = 1 |
| | TGT0_ADD = 3Dh | // Magnetometer output registers start from 68h |
| | TGT0_SUBADD = 68h | // Set up a continuous 6-byte read from the I²C controller interface |
| | TGT0_CONFIG = 80h \| 06h | |

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

In order to limit intervention by the host processor and facilitate postprocessing data for event recognition, the LSM6DSV80X embeds a 1.5 KB (up to 4.5 KB with the compression feature enabled) first-in, first-out buffer (FIFO).

The FIFO can be configured to store the following data:

- Gyroscope sensor data
- Accelerometer sensor data (either low-*g* or high-*g*)
- Timestamp data
- Temperature sensor data
- External sensor (connected to sensor hub interface) data
- Step counter (and associated timestamp) data
- SFLP game rotation vector, gravity vector, gyroscope bias
- High-*g* accelerometer peak value
- Machine learning core filters, features, and results
- Finite state machine events

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

- Tag, 1 byte
- Data, 6 bytes

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

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

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

- Bypass mode
- FIFO mode
- Continuous mode
- Continuous-to-FIFO mode
- Bypass-to-continuous mode
- Bypass-to-FIFO mode
- ContinuousWTM-to-full mode

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

A programmable FIFO threshold can be set in FIFO_CTRL1 using the WTM_[7:0] 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_TH and INT1_FIFO_OVR bits of the INT1_CTRL register, and through the INT2_ FIFO_FULL, INT2_FIFO_TH and INT2_FIFO_OVR bits of the INT2_CTRL register.

Finally, FIFO embeds a compression algorithm that the user can enable in order to have up to 4.5 KB data stored in FIFO and take advantage in terms of interface communication length for FIFO flushing and communication power consumption.

## 8.1 FIFO description and batched sensors

FIFO is divided into 256 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 1792 bytes and can contain 1536 bytes of data. The TAG byte contains the information indicating which data is stored in the FIFO data field and other useful information.

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

Moreover, in order to increase its capability, the FIFO embeds a compression algorithm for the low-*g* accelerometer and gyroscope data (refer to Section 8.10: FIFO compression for further details).

Batched sensors can be classified in three different categories:

- Main sensors, which are physical sensors:
  - Accelerometer sensor (either low-*g* or high-*g*)
  - Gyroscope sensor
- Auxiliary sensors, which contain information of the status of the device:
  - Timestamp sensor
  - Configuration-change sensor (CFG-change)
  - Temperature sensor
- Virtual sensors:
  - External sensors read from sensor hub interface
  - Step counter sensor
  - SFLP game rotation vector, gravity vector, and gyroscope bias
  - High-*g* accelerometer peak value
  - Machine learning core filters, features, and results
  - Finite state machine events

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

A write to FIFO can be triggered by the following different events:

- Internal data-ready signal (fastest sensor between the low-*g* accelerometer, the high-*g* accelerometer, and gyroscope)
- Sensor hub data-ready
- Step detection event
- Virtual sensor new data available

## 8.2 FIFO registers

The FIFO buffer is managed by:

- Six control registers: FIFO_CTRL1, FIFO_CTRL2, FIFO_CTRL3, FIFO_CTRL4, COUNTER_BDR_REG1, COUNTER_BDR_REG2
- Two status registers: FIFO_STATUS1 and FIFO_STATUS2
- Seven output registers (tag + data): FIFO_DATA_OUT_TAG, FIFO_DATA_OUT_X_L, FIFO_DATA_OUT_X_H, FIFO_DATA_OUT_Y_L, FIFO_DATA_OUT_Y_H, FIFO_DATA_OUT_Z_L, FIFO_DATA_OUT_Z_H
- Some additional bits to route FIFO events to the two interrupt lines: INT1_CNT_BDR, INT1_FIFO_FULL, INT1_FIFO_OVR, INT1_FIFO_TH bits of the INT1_CTRL register and INT2_CNT_BDR, INT2_FIFO_FULL, INT2_FIFO_OVR, INT2_FIFO_TH bits of the INT2_CTRL register
- Some additional bits for other features:
  - FIFO_COMPR_EN bit of the EMB_FUNC_EN_B embedded function register in order to enable the FIFO compression algorithm
  - STEP_COUNTER_FIFO_EN bit of the EMB_FUNC_FIFO_EN_A register in order to enable batching the step counter data in FIFO
  - MLC_FIFO_EN bit of the EMB_FUNC_FIFO_EN_A register in order to enable batching the machine learning core results in FIFO
  - MLC_FILTER_FEATURE_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register in order to enable batching the machine learning core filters and features in FIFO
  - FSM_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register in order to enable batching the finite state machine events in FIFO
  - SFLP_GBIAS_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register in order to enable batching the gyroscope bias data in FIFO (the SFLP embedded function must be enabled)
  - SFLP_GRAVITY_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register in order to enable batching the gravity vector data in FIFO (the SFLP embedded function must be enabled)
  - SFLP_GAME_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register in order to enable batching the game rotation vector data in FIFO (the SFLP embedded function must be enabled)
  - FIFO_COMPR_INIT bit of the EMB_FUNC_INIT_B embedded function register in order to request a reinitialization of the FIFO compression algorithm
  - BATCH_EXT_SENS_0_EN, BATCH_EXT_SENS_1_EN, BATCH_EXT_SENS_2_EN, BATCH_EXT_SENS_3_EN bits of the TGT0_CONFIG, TGT1_CONFIG, TGT2_CONFIG, TGT3_CONFIG sensor hub registers, which enable batching the related external sensor data in FIFO

### 8.2.1 FIFO_CTRL1

The FIFO_CTRL1 register contains the FIFO watermark threshold level. The value of 1 LSB of the FIFO threshold level is referred to as a FIFO word (7 bytes).

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

In order to limit the FIFO depth to the watermark level, the STOP_ON_WTM bit must be set to 1 in the FIFO_CTRL2 register. If STOP_ON_WTM = 1, the watermark threshold level must be set equal to or greater than double the maximum number of sensors batched in FIFO.

**Table 68. FIFO_CTRL1 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| WTM_7 | WTM_6 | WTM_5 | WTM_4 | WTM_3 | WTM_2 | WTM_1 | WTM_0 |

### 8.2.2 FIFO_CTRL2

**Table 69. FIFO_CTRL2 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| STOP_ ON_WTM | FIFO_COMPR _RT_EN | 0 | ODR_CHG _EN | 0 | UNCOMPR _RATE_1 | UNCOMPR _RATE_0 | 0 |

The FIFO_CTRL2 register contains the bit STOP_ON_WTM, which allows limiting the FIFO depth to the watermark level.

The FIFO_CTRL2 register also contains the bits to manage the FIFO compression algorithm for the low-*g* accelerometer and gyroscope sensors:

- FIFO_COMPR_RT_EN bit allows runtime enabling / disabling of the compression algorithm: if the bit is set to 1, the compression is enabled, otherwise it is disabled.
- UNCOMPR_RATE_[1:0] configures the compression algorithm to write noncompressed data at a specific rate. The following table summarizes possible configurations.

**Table 70. Forced noncompressed data write configurations**

| UNCOPTR_RATE[1:0] | Forced noncompressed data writes |
|---|---|
| 00 | Never |
| 01 | Every 8 batch data rate |
| 10 | Every 16 batch data rate |
| 11 | Every 32 batch data rate |

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

### 8.2.3 FIFO_CTRL3

Table 71. **FIFO_CTRL3 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| BDR_GY_3 | BDR_GY_2 | BDR_GY_1 | BDR_GY_0 | BDR_XL_3 | BDR_XL_2 | BDR_XL_1 | BDR_XL_0 |

The FIFO_CTRL3 register contains the fields to select the write frequency in FIFO for the low-*g* accelerometer and gyroscope sensor data. The selected batch data rate must be equal to or lower than the output data rate configured through the ODR_XL and ODR_G fields of the CTRL1_XL and CTRL2_G registers.

The following tables indicate all the selectable batch data rates.

Table 72. **Low-*g* accelerometer batch data rate**

| BDR_XL[3:0] | Batch data rate [Hz] |
|---|---|
| 0000 | Not batched in FIFO |
| 0001 | 1.875 |
| 0010 | 7.5 |
| 0011 | 15 |
| 0100 | 30 |
| 0101 | 60 |
| 0110 | 120 |
| 0111 | 240 |
| 1000 | 480 |
| 1001 | 960 |
| 1010 | 1920 |
| 1011 | 3840 |
| 1100 | 7680 |

Table 73. **Gyroscope batch data rate**

| BDR_GY[3:0] | Batch data rate [Hz] |
|---|---|
| 0000 | Not batched in FIFO |
| 0001 | 1.875 |
| 0010 | 7.5 |
| 0011 | 15 |
| 0100 | 30 |
| 0101 | 60 |
| 0110 | 120 |
| 0111 | 240 |
| 1000 | 480 |
| 1001 | 960 |
| 1010 | 1920 |
| 1011 | 3840 |
| 1100 | 7680 |

### 8.2.4 FIFO_CTRL4

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

The timestamp write rate is configured as the maximum batch data rate (BDR_MAX) divided by the decimation factor specified in the DEC_TS_BATCH_[1:0] field. BDR_MAX is the maximum batch data rate among the following batch data rates:

- Low-*g* accelerometer batch data rate (BDR_XL)
- Gyroscope batch data rate (BDR_GY)
- Sensor hub batch data rate (BDR_SHUB)
- High-*g* accelerometer batch data rate (equal to ODR_XL_HG), if batching the high-*g* accelerometer data in FIFO is enabled

The programmable decimation factors are indicated in the table below.

**Table 74. Timestamp batch data rate**

| DEC_TS_BATCH_[1:0] | Timestamp batch data rate [Hz] |
|---|---|
| 00 | Not batched in FIFO |
| 01 | BDR_MAX |
| 10 | BDR_MAX / 8 |
| 11 | BDR_MAX / 32 |

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

**Table 75. Temperature sensor batch data rate**

| ODR_T_BATCH_[1:0] | Temperature batch data rate [Hz] |
|---|---|
| 00 | Not batched in FIFO |
| 01 | 1.875 |
| 10 | 15 |
| 11 | 60 |

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

**Table 76. FIFO_CTRL4 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| DEC_TS_BATCH_1 | DEC_TS_BATCH_0 | ODR_T_BATCH_1 | ODR_T_BATCH_0 | 0 | FIFO_MODE2 | FIFO_MODE1 | FIFO_MODE0 |

### 8.2.5 COUNTER_BDR_REG1

Since the FIFO might contain metainformation (that is, CFG-change sensor) and low-*g* accelerometer and gyroscope data might be compressed, the FIFO provides a way to synchronize the FIFO reading on the basis of the low-*g* accelerometer or gyroscope actual number of samples stored in FIFO: the BDR counter.

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

**Table 77. COUNTER_BDR_REG1 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| 0 | TRIG_ COUNTER_BDR_1 | TRIG_ COUNTER_BDR_0 | 0 | XL_HG_BATCH _EN | 0 | CNT_BDR _TH_9 | CNT_BDR _TH_8 |

The TRIG_COUNTER_BDR_[1:0] field selects the trigger for the BDR counter:

- 00: low-*g* accelerometer sensor is selected as the trigger
- 01: gyroscope sensor is selected as the trigger
- 10: reserved
- 11: high-*g* accelerometer sensor is selected as the trigger

The user can select the threshold that generates the COUNTER_BDR_IA event in the FIFO_STATUS2 register. Once the internal BDR counter reaches the threshold, the COUNTER_BDR_IA bit is set to 1. The threshold is configurable through the CNT_BDR_TH_[9:0] bits. The upper part of the field is contained in register COUNTER_BDR_REG1. 1 LSB value of the CNT_BDR_TH threshold level is referred to as one low-*g* accelerometer/gyroscope sample (X, Y, and Z data).

The BDR counter is automatically reset when the FIFO is empty.

The user can enable batching the high-*g* accelerometer by setting the XL_HG_BATCH_EN bit to 1.

*Note: The high-g accelerometer BDR is not configurable and is always equal to the configured ODR.*

### 8.2.6 COUNTER_BDR_REG2

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

**Table 78. COUNTER_BDR_REG2 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| CNT_BDR _TH_7 | CNT_BDR _TH_6 | CNT_BDR _TH_5 | CNT_BDR _TH_4 | CNT_BDR _TH_3 | CNT_BDR _TH_2 | CNT_BDR _TH_1 | CNT_BDR _TH_0 |

### 8.2.7 FIFO_STATUS1

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

**Table 79. FIFO_STATUS1 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| DIFF_ FIFO_7 | DIFF_ FIFO_6 | DIFF_ FIFO_5 | DIFF_ FIFO_4 | DIFF_ FIFO_3 | DIFF_ FIFO_2 | DIFF_ FIFO_1 | DIFF_ FIFO_0 |

### 8.2.8 FIFO_STATUS2

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

Table 80. **FIFO_STATUS2 register**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| FIFO_ WTM_IA | FIFO_ OVR_IA | FIFO_ FULL_IA | COUNTER _BDR_IA | FIFO_OVR _LATCHED | 0 | 0 | DIFF_ FIFO_8 |

- FIFO_WTM_IA represents the watermark status. This bit goes high when the number of FIFO words (7 bytes each) already stored in the FIFO is equal to or higher than the watermark threshold level. The watermark status signal can be driven to the two interrupt pins by setting the INT1_FIFO_TH bit of the INT1_CTRL register or the INT2_FIFO_TH bit of the INT2_CTRL register to 1.
- FIFO_OVR_IA goes high when the FIFO is completely filled and at least one sample has already been overwritten to store the new data. This signal can be driven to the two interrupt pins by setting the INT1_FIFO_OVR bit of the INT1_CTRL register or the INT2_FIFO_OVR bit of the INT2_CTRL register to 1.
- FIFO_FULL_IA goes high when the next set of data that is stored in FIFO makes the FIFO completely full (that is, DIFF_FIFO_8 = 1) or generate a FIFO overrun. This signal can be driven to the two interrupt pins by setting the INT1_FIFO_FULL bit of the INT1_CTRL register or the INT2_FIFO_FULL bit of the INT2_CTRL register to 1.
- COUNTER_BDR_IA represents the BDR-counter status. This bit goes high when the number of accelerometer or gyroscope batched samples (on the base of the selected sensor trigger) reaches the BDR-counter threshold level configured through the CNT_BDR_TH_[9:0] bits of the COUNTER_BDR_REG1 and COUNTER_BDR_REG2 registers. The COUNTER_BDR_IA bit is automatically reset when the FIFO_STATUS2 register is read. The BDR-counter status can be driven to the two interrupt pins by setting the INT1_CNT_BDR bit of the INT1_CTRL register or the INT2_CNT_BDR bit of the INT2_CTRL register to 1.
- FIFO_OVR_LATCHED, as FIFO_OVR_IA, goes high when the FIFO is completely filled and at least one sample has already been overwritten to store the new data. The difference between the two flags is that FIFO_OVR_LATCHED is reset when the FIFO_STATUS2 register is read, whereas the FIFO_OVR_IA is reset when at least one FIFO word is read. This allows detecting a FIFO overrun condition during reading data from FIFO.
- DIFF_FIFO_8 contains the upper part of the number of unread words stored in the FIFO. The lower part is represented by the DIFF_FIFO_[7:0] bits in FIFO_STATUS1. The value of the DIFF_FIFO_[8:0] field corresponds to the number of 7-byte words in the FIFO.

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

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

### 8.2.9 FIFO_DATA_OUT_TAG

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

**Table 81. FIFO_DATA_OUT_TAG register**

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

- TAG_SENSOR_[4:0] field identifies the sensors stored in the 6 data bytes (Table 82).
- TAG_CNT_[1:0] field identifies the FIFO time slot (described in the next sections).

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

**Table 82. TAG_SENSOR field and associated sensor**

| TAG_SENSOR_[4:0] | Sensor name | Sensor category | Description |
|---|---|---|---|
| 0x00 | Empty | - | FIFO empty condition |
| 0x01 | Gyroscope NC | Main | Gyroscope uncompressed data |
| 0x02 | Accelerometer NC | Main | Low-*g* accelerometer uncompressed data |
| 0x03 | Temperature | Auxiliary | Temperature data |
| 0x04 | Timestamp | Auxiliary | Timestamp data |
| 0x05 | CFG_Change | Auxiliary | Metainformation data |
| 0x06 | Accelerometer NC_T_2 | Main | Low-*g* accelerometer uncompressed batched at two times the previous time slot |
| 0x07 | Accelerometer NC_T_1 | Main | Low-*g* accelerometer uncompressed data batched at the previous time slot |
| 0x08 | Accelerometer 2xC | Main | Low-*g* accelerometer 2x compressed data |
| 0x09 | Accelerometer 3xC | Main | Low-*g* accelerometer 3x compressed data |
| 0x0A | Gyroscope NC_T_2 | Main | Gyroscope uncompressed data batched at two times the previous time slot |
| 0x0B | Gyroscope NC_T_1 | Main | Gyroscope uncompressed data batched at the previous time slot |
| 0x0C | Gyroscope 2xC | Main | Gyroscope 2x compressed data |
| 0x0D | Gyroscope 3xC | Main | Gyroscope 3x compressed data |
| 0x0E | Sensor Hub Target 0 | Virtual | Sensor hub data from target 0 |
| 0x0F | Sensor Hub Target 1 | Virtual | Sensor hub data from target 1 |
| 0x10 | Sensor Hub Target 2 | Virtual | Sensor hub data from target 2 |
| 0x11 | Sensor Hub Target 3 | Virtual | Sensor hub data from target 3 |
| 0x12 | Step counter | Virtual | Step counter data |
| 0x13 | Game rotation vector | Virtual | SFLP-generated game rotation vector |
| 0x16 | Gyroscope bias | Virtual | SFLP-generated gyroscope bias |
| 0x17 | Gravity vector | Virtual | SFLP-generated gravity vector |
| 0x18 | High-*g* accelerometer peak | Virtual | High-*g* accelerometer peak value |
| 0x19 | Sensor hub nack | Virtual | Sensor hub nack from target 0/1/2/3 |
| 0x1A | MLC result | Virtual | Machine learning core generated result |
| 0x1B | MLC filter | Virtual | Machine learning core generated filter |
| 0x1C | MLC feature | Virtual | Machine learning core generated feature |
| 0x1D | High-*g* accelerometer | Main | High-*g* accelerometer data |
| 0x1F | FSM events | Virtual | FSM long counter timeout, long counter value, and detected motion pattern |

### 8.2.10 FIFO_DATA_OUT

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

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

## 8.3 FIFO batched sensors

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

- Main sensors
- Auxiliary sensors
- Virtual sensors

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

## 8.4 Main sensors

The main sensors are the physical sensors of the LSM6DSV80X device: low-*g* accelerometer, high-*g* accelerometer, and gyroscope. The batch data rate can be configured through the BDR_XL_[3:0] and BDR_GY_[3:0] fields of the FIFO_CTRL3 register, for the low-*g* accelerometer and gyroscope, respectively. The batch data rate must be equal to or lower than the related sensor output data rate configured through the ODR_XL_[3:0] and ODR_G_[3:0] fields of the CTRL1 and CTRL2 registers.

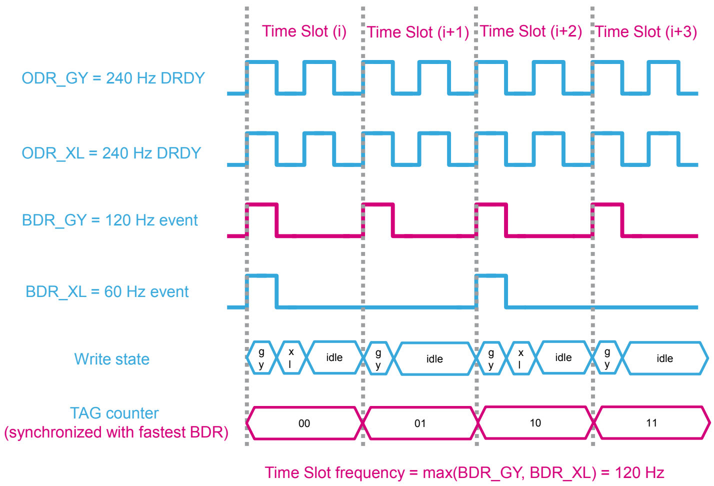
Batching the high-*g* accelerometer data can be enabled by setting the XL_HG_BATCH_EN bit to 1 in the COUNTER_BDR_REG1 register. The high-*g* accelerometer data are stored in FIFO according to the ODR_XL_HG[2:0] field of the CTRL1_XL_HG register.

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

An example of a batch data rate event is shown in Figure 19. Main sensors and time slot definitions. The BDR_GY event and BDR_XL event identify the time in which the corresponding sensor data is written to the FIFO. The evolution of the TAG counter identifies different time slots and its frequency is equivalent to the maximum value between BDR_XL and BDR_GY, since high-*g* accelerometer data are not batched in this example.

In the general case, the frequency of the TAG counter is equivalent to the maximum batch data rate of either the low-*g* accelerometer, high-*g* accelerometer, or gyroscope, whichever is faster.

**Figure 19. Main sensors and time slot definitions**




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

**Table 83. Main sensors output data format in FIFO**

| TAG | X_L | X_H | Y_L | Y_H | Z_L | Z_H |
|---|---|---|---|---|---|---|

## 8.5 Auxiliary sensors

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

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

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

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

**Table 84. Temperature output data format in FIFO**

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

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

**Table 85. Timestamp output data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| TIMESTAMP[7:0] | FIFO_DATA_OUT_X_L |
| TIMESTAMP[15:8] | FIFO_DATA_OUT_X_H |
| TIMESTAMP[23:16] | FIFO_DATA_OUT_Y_L |
| TIMESTAMP[31:24] | FIFO_DATA_OUT_Y_H |
| BDR_SHUB | FIFO_DATA_OUT_Z_L[3:0] |
| XL_HG_BATCH_EN | FIFO_DATA_OUT_Z_L[5] |
| 0 | FIFO_DATA_OUT_Z_L[7:6] |
| BDR_XL | FIFO_DATA_OUT_Z_H[3:0] |
| BDR_GY | FIFO_DATA_OUT_Z_H[7:4] |

As shown in Table 85, timestamp data also contain some metainformation, which can be used to detect a BDR change if the CFG-Change sensor is not batched in FIFO: the batch data rate of both the main sensors and the sensor hub. BDR_SHUB cannot be configured through a dedicated register. It is the result of the configured sensor hub ODR through the SHUB_ODR_[2:0] bits of the TGT0_CONFIG sensor hub register and the effective trigger sensor output data rate (the fastest between the low-*g* accelerometer or gyroscope if the internal trigger is used). For the complete description of BDR_SHUB, refer to the next section about virtual sensors.

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

This sensor can be used to correlate data from the sensors to the device timestamp without storing the timestamp each time. It could also be used 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 86. CFG-Change output data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| OP_MODE_XL | FIFO_DATA_OUT_X_L[2:0] |
| OP_MODE_G | FIFO_DATA_OUT_X_L[6:4] |
| LPF1_G_EN | FIFO_DATA_OUT_X_L[7] |
| LPF1_G_BW | FIFO_DATA_OUT_X_H[2:0] |
| FS_G_[2:0] | FIFO_DATA_OUT_X_H[7:5] |
| LPF2_XL_EN | FIFO_DATA_OUT_Y_L[0] |
| HP_LPF2_XL_BW | FIFO_DATA_OUT_Y_L[3:1] |
| FS_XL | FIFO_DATA_OUT_Y_L[7:6] |
| BDR_SHUB | FIFO_DATA_OUT_Y_H[3:0] |
| XL_HG_BATCH_EN | FIFO_DATA_OUT_Y_H[5] |
| Gyroscope startup(1) | FIFO_DATA_OUT_Y_H[6] |
| FIFO_COMPR_RT_EN | FIFO_DATA_OUT_Y_H[7] |
| ODR_XL | FIFO_DATA_OUT_Z_L[3:0] |
| ODR_G | FIFO_DATA_OUT_Z_L[7:4] |
| BDR_XL | FIFO_DATA_OUT_Z_H[3:0] |
| BDR_GY | FIFO_DATA_OUT_Z_H[7:4] |

1. *Internal signal that is set to 0 when the gyroscope finishes the startup phase (maximum startup time is 70 ms).*

## 8.6 Virtual sensors

Virtual sensors are divided into the following categories:

- External sensors, read from the sensor hub interface
- Step counter sensor
- SFLP-generated sensors
- High-*g* accelerometer peak value sensor
- MLC-generated sensors
- FSM-generated sensors

### 8.6.1 External sensors and nack sensor

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

They are continuous virtual sensors with the batch data rate (BDR_SHUB) corresponding to the current value of the SHUB_ODR_[2:0] field in the TGT0_CONFIG register, if an internal trigger is used (sensor hub read triggered by the low-*g* accelerometer or gyroscope data-ready signal). This value is limited by the effective trigger sensor output data rate (the fastest between the low-*g* accelerometer or gyroscope). If external sensors are not batched or an external trigger is used, BDR_SHUB is set to 0. The following table shows the possible values of the BDR_SHUB field.

Table 87. **BDR_SHUB**

| BDR_SHUB | BDR [Hz] |
|---|---|
| 0000 | Not batched or external trigger used |
| 0001 | 1.875 |
| 0010 | 7.5(1) |
| 0011 | 15 |
| 0100 | 30 |
| 0101 | 60 |
| 0110 | 120 |
| 0111 | 240 |
| 1000 | 480 |

1. *This value can be obtained by selecting SHUB_ODR_[2:0] different from 000 and using an internal trigger (low-g accelerometer or gyroscope) with ODR equal to 7.5 Hz.*

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

It is possible to selectively enable batching the data of the different external sensors using the BATCH_EXT_SENS_0_EN, BATCH_EXT_SENS_1_EN, BATCH_EXT_SENS_2_EN, BATCH_EXT_SENS_3_EN bits of the TGT0_CONFIG, TGT1_CONFIG, TGT2_CONFIG, TGT3_CONFIG sensor hub registers. It is recommended to set these bits to 0 if the respective target is not configured for reading or is not active.

*Note: If external sensors are batched in FIFO, batching auxiliary sensors is not supported in bypass-to-continuous and bypass-to-FIFO modes.*

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

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

**Table 88. Nack sensor output data format in FIFO**

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

### 8.6.2 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 batching step counter data (set the STEP_COUNTER_FIFO_EN bit to 1 in the EMB_FUNC_FIFO_EN_A embedded functions register)

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

**Table 89. 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 |

### 8.6.3 SFLP-generated sensors

A dedicated sensor fusion block (SFLP) is available for generating the following virtual sensors based on processing the accelerometer and gyroscope data:

- Game rotation vector, which provides a quaternion representing the attitude of the device
- Gravity vector, which provides a three-dimensional vector representing the direction of gravity
- Gyroscope bias, which provides a three-dimensional vector representing the gyroscope bias

SFLP-generated sensors are read only from FIFO and they are selectively enabled:

- Game rotation vector is batched by setting the SFLP_GAME_FIFO_EN bit of the EMB_FUNC_FIFO_EN_A register to 1.
- Gravity vector is batched by setting the SFLP_GRAVITY_FIFO_EN bit of EMB_FUNC_FIFO_EN_A register to 1.
- Gyroscope bias is batched by setting the SFLP_GBIAS_FIFO_EN bit of the EMB_FUNC_FIFO_EN_A register to 1.

If batching in FIFO is enabled, the SFLP-generated sensors are stored in FIFO according to the SFLP output data rate.

The format for the SFLP-generated sensors in FIFO is listed below:

- Game rotation vector: X, Y, Z axes (vector part of the quaternion), and w (scalar part of the quaternion) are stored in half-precision floating-point format. In particular, every time that a value of the game rotation vector is batched two subsequent FIFO entries (tag + FIFO data) tagged with 0x13 are stored following the format described in Table 90 and Table 91. The FIFO_DATA_OUT_X is used to identify the two 32-bit sets that compose the complete vector.
- Gravity vector: X, Y, and Z axes are stored as 16-bit two's complement number with ±2 *g* sensitivity.
- Gyroscope bias: X, Y, and Z axes are stored as 16-bit two's complement number with ±125 dps sensitivity.

**Table 90. Game rotation vector data format in FIFO (first word)**

| SFLP_GAME_ROTATION_VECTOR | FIFO_DATA_OUT registers |
|---|---|
| 0x00 | FIFO_DATA_OUT_X_L |
| 0x00 | FIFO_DATA_OUT_X_H |
| SFLP_QUATX_L | FIFO_DATA_OUT_Y_L |
| SFLP_QUATX_H | FIFO_DATA_OUT_Y_H |
| SFLP_QUATY_L | FIFO_DATA_OUT_Z_L |
| SFLP_QUATY_H | FIFO_DATA_OUT_Z_H |

**Table 91. Game rotation vector data format in FIFO (second word)**

| SFLP_GAME_ROTATION_VECTOR | FIFO_DATA_OUT registers |
|---|---|
| 0x01 | FIFO_DATA_OUT_X_L |
| 0x00 | FIFO_DATA_OUT_X_H |
| SFLP_QUATZ_L | FIFO_DATA_OUT_Y_L |
| SFLP_QUATZ_H | FIFO_DATA_OUT_Y_H |
| SFLP_QUATW_L | FIFO_DATA_OUT_Z_L |
| SFLP_QUATW_H | FIFO_DATA_OUT_Z_H |

### 8.6.4 High-*g* accelerometer peak value sensor

The LSM6DSV80X finite state machine supports storing the peak-tracking output data in the FIFO buffer every time a specific FSM command, needed to stop the peak-tracking monitoring, is issued. Refer to the finite state machine application note available on www.st.com for details about FSM commands and configuration.

The peak-tracking output data are identified by a FIFO tag value readable from the FIFO_DATA_OUT_TAG register equal to 18h. The 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, and FIFO_DATA_OUT_Z_H registers contain the X, Y, and Z high-*g* accelerometer data in half-precision floating-point format in [h*g*] by default. The format of peak-tracking output in FIFO is indicated in the following table.

**Table 92. High-*g* accelerometer peak value data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| X_L | FIFO_DATA_OUT_X_L |
| X_H | FIFO_DATA_OUT_X_H |
| Y_L | FIFO_DATA_OUT_Y_L |
| Y_H | FIFO_DATA_OUT_Y_H |
| Z_L | FIFO_DATA_OUT_Z_L |
| Z_H | FIFO_DATA_OUT_Z_H |

### 8.6.5 MLC-generated sensors

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

- Results
- Filters
- Features, including windowed and recursive 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_A 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 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_B register must be set to 1 to globally enable storing MLC filters or features in FIFO.

MLC filters are stored in FIFO at a rate equivalent to the MLC output data rate (MLC_ODR 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.

Batching MLC 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_B register must be set to 1 to globally enable storing MLC filters or features in FIFO.

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

MLC recursive features (like MLC filters) are stored in FIFO at a rate equivalent to the MLC output data rate (MLC_ODR).

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

**Table 93. MLC results data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| MLCx_SRC | FIFO_DATA_OUT_X_L |
| Index of MLC_SRC[1] | 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 |

1. *MLCx_SRC registers are indexed from 0 to 3 (for example, MLC1_SRC is indexed as 0).*

**Table 94. MLC filters or features data format 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.*

### 8.6.6 FSM-generated sensors

The following finite state machine (FSM-generated) virtual sensors can be stored in FIFO:

- Long counter timeout event
- Long counter value event
- Detected motion pattern event

In order to store FSM-generated sensors in FIFO, the FSM block must be enabled by setting the FSM_EN bit in the EMB_FUNC_EN_B register and one or more FSM algorithms must be enabled by setting the corresponding FSMx_EN bit of the FSM_ENABLE register.

Batching FSM events is enabled by setting the FSM_FIFO_EN bit of the EMB_FUNC_FIFO_EN_B register to 1.

All the FSM events batched in FIFO have the same value (1Fh) for the FIFO_DATA_OUT_TAG register, and the user can distinguish between them by looking at the value of the FIFO_DATA_OUT_X_H register. Refer to Table 95. FSM long counter timeout data format in FIFO, Table 96. FSM long counter value (L) data format in FIFO, Table 97. FSM long counter value (H) data format in FIFO, and Table 98. FSM-detected motion pattern data format in FIFO for details about FSM events in FIFO data format.

An FSM long counter timeout event data contains the timestamp value corresponding to the FSM long counter timeout event, generated every time that the long counter value reaches the configured timeout value. Refer to the finite state machine application note available on www.st.com for details about FSM commands and configuration.

An FSM long counter value event data contains the actual long counter value and the timestamp value corresponding to the FSM long counter value event, generated every time that the corresponding FSM command is issued. The FSM_LONG_COUNTER_L and FSM_LONG_COUNTER_H bytes are stored in two different FIFO words. Refer to the finite state machine application note available on www.st.com for details about FSM commands and configuration.

*Note: The high part of the long counter value, identified by the FIFO_DATA_OUT_X_H register value equal to 0Ah, is stored in FIFO only if the FSM_LONG_COUNTER_H register value is not zero.*

An FSM-detected motion pattern event data contains the FSM_OUTSx value, the index of the FSM, and the timestamp value corresponding to the FSM-detected motion pattern event, generated every time that a target motion pattern is detected by an FSM. Refer to the finite state machine application note available on www.st.com for details about FSM commands and configuration.

The format of FSM events in FIFO is indicated in the following tables.

**Table 95. FSM long counter timeout data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| 0x00 | FIFO_DATA_OUT_X_L |
| 0x00 | 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 96. FSM long counter value (L) data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| FSM_LONG_COUNTER_L | FIFO_DATA_OUT_X_L |
| 0x09 | 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 97. **FSM long counter value (H) data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| FSM_LONG_COUNTER_H | FIFO_DATA_OUT_X_L |
| 0x0A | 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 98. **FSM-detected motion pattern data format in FIFO**

| Data | FIFO_DATA_OUT registers |
|---|---|
| OUTSx | FIFO_DATA_OUT_X_L |
| Index of the FSM(1) | 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 |

1. *FSMs are indexed from 1 to 8.*

## 8.7 FIFO modes

The LSM6DSV80X FIFO buffer can be configured to operate in seven different modes, selectable through the FIFO_MODE_[2:0] field of the FIFO_CTRL4 register. The available configurations ensure a high level of flexibility and extend the number of functions usable in application development.

Bypass, FIFO, continuous, continuous-to-FIFO, bypass-to-continuous, bypass-to-FIFO, and continuousWTM-to-full modes are described in the following paragraphs.

### 8.7.1 Bypass mode

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

### 8.7.2 FIFO mode

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

Follow these steps for FIFO mode configuration:

1. Enable the sensor data to be stored in FIFO with the corresponding batch data rate (if configurable).
2. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 001 to enable FIFO mode.

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

When the FIFO is full, the DIFF_FIFO_8 bit of the FIFO_STATUS2 register is set to 1 and no more data are stored in the FIFO buffer. Data can be retrieved by reading all the FIFO_DATA_OUT (from 78h to 7Eh) registers for the number of times specified by the DIFF_FIFO_[8:0] bits of the FIFO_STATUS1 and FIFO_STATUS2 registers.

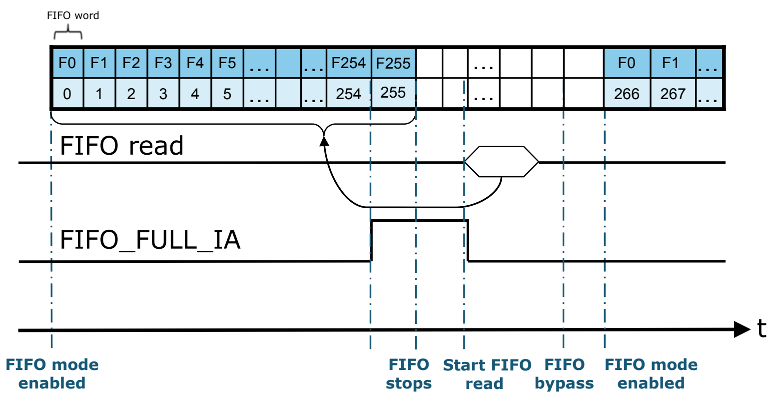
Using the FIFO_WTM_IA bit of the FIFO_STATUS2 register, data can also be retrieved when a threshold level (WTM_[7:0] in the FIFO_CTRL1 register) is reached if the application requires a lower number of samples in the FIFO.

If the STOP_ON_WTM bit of the FIFO_CTRL2 register is set to 1, the FIFO size is limited to the value of the WTM_[7:0] bits in the FIFO_CTRL1 register. In this case, the FIFO_FULL_IA bit of the FIFO_STATUS2 register is set high when the number of samples in FIFO reaches or exceeds the WTM_[7:0] value on the next FIFO write operation.

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

Figure 20. FIFO mode (STOP_ON_WTM = 0) shows an example of FIFO mode usage; the data from just one sensor are stored in the FIFO. In these conditions, the number of samples that can be stored in the FIFO buffer is 256 (with compression algorithm disabled). The FIFO_FULL_IA bit of the FIFO_STATUS2 register goes high just after the level labeled as 254 to notify that the FIFO buffer will be completely filled at the next FIFO write operation. After the FIFO is full (FIFO_DIFF_8 = 1), the data collection stops.

**Figure 20. FIFO mode (STOP_ON_WTM = 0)**



### 8.7.3 Continuous mode

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

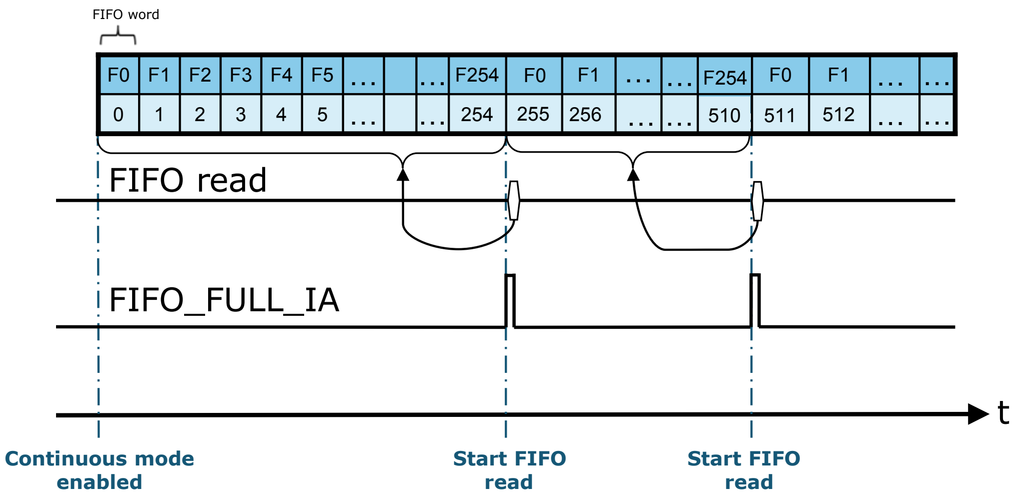
Follow these steps for continuous mode configuration (if the low-*g* accelerometer/high-*g* accelerometer/gyroscope data-ready is used as the FIFO trigger):

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

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

Figure 21. Continuous mode shows an example of the continuous mode usage. In the example, data from just one sensor are stored in the FIFO and the FIFO samples are read on the FIFO_FULL_IA event and faster than 1 * ODR so that no data is lost. In these conditions, the number of samples stored is 255.

**Figure 21. Continuous mode**



### 8.7.4 Continuous-to-FIFO mode

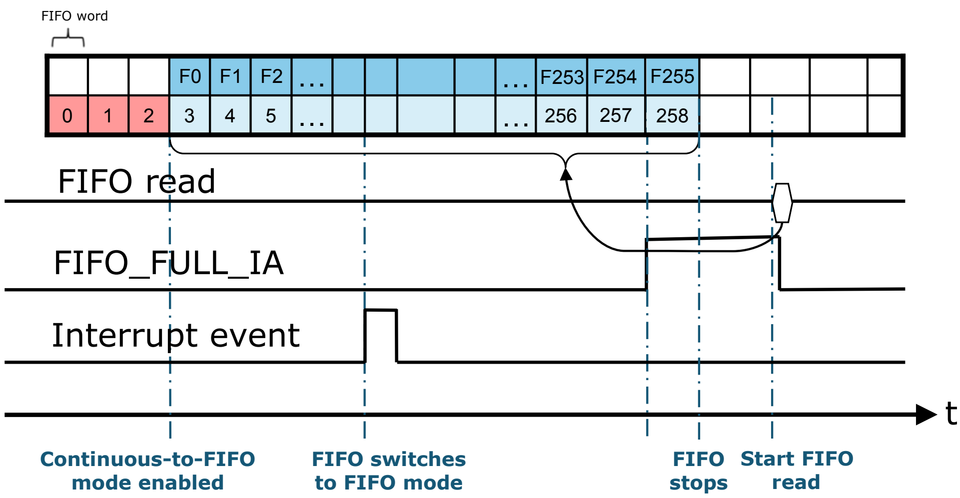
This mode is a combination of the continuous and FIFO modes previously described. In continuous-to-FIFO mode, the FIFO buffer starts operating in continuous mode and switches to FIFO mode when an event condition occurs.

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1.
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1.
- Free-fall: event detection has to be configured and the INT2_FF bit of the MD2_CFG register has to be set to 1.
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1.
- 6D: event detection has to be configured and the INT2_6D bit of the MD2_CFG register has to be set to 1.
- High-*g* wake-up: event detection has to be configured and the INT2_HG_WU bit of the HG_FUNCTIONS_ENABLE register has to be set to 1.

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

**Figure 22. Continuous-to-FIFO mode**




Follow these steps for continuous-to-FIFO mode configuration (if the low-*g* accelerometer/high-*g* accelerometer/ gyroscope data-ready is used as the FIFO trigger):

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

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

If the STOP_ON_WTM bit of the FIFO_CTRL2 register is set to 1, the FIFO size is limited to the value of the WTM_[7:0] bits in the FIFO_CTRL1 register. In this case, the FIFO_FULL_IA bit of the FIFO_STATUS2 register goes high when the number of samples in FIFO reaches or exceeds the WTM_[7:0] value at the next FIFO write operation.

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

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

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

### 8.7.5 Bypass-to-continuous mode

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

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1.
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1.
- Free-fall: event detection has to be configured and the INT2_FF bit of the MD2_CFG register has to be set to 1.
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1.
- 6D: event detection has to be configured and the INT2_6D bit of the MD2_CFG register has to be set to 1.
- High-*g* wake-up: event detection has to be configured and the INT2_HG_WU bit of the HG_FUNCTIONS_ENABLE register has to be set to 1.

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

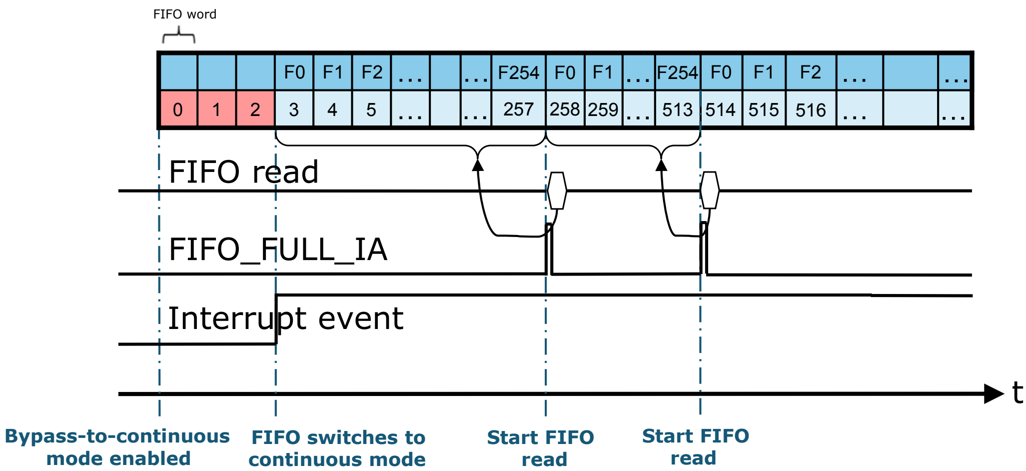
Follow these steps for bypass-to-continuous mode configuration (if the low-*g* accelerometer/high-*g* accelerometer/gyroscope data-ready is used as the FIFO trigger):

1. Configure one of the events as previously described.
2. Enable the sensor data to be stored in FIFO with the corresponding batch data rate (if configurable).
3. Set the FIFO_MODE_[2:0] bits in the FIFO_CTRL4 register to 100 to enable FIFO bypass-to-continuous mode.

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

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

**Figure 23. Bypass-to-continuous mode**



### 8.7.6 Bypass-to-FIFO mode

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

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1.
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1.
- Free-fall: event detection has to be configured and the INT2_FF bit of the MD2_CFG register has to be set to 1.
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1.
- 6D: event detection has to be configured and the INT2_6D bit of the MD2_CFG register has to be set to 1.
- High-*g* wake-up: event detection has to be configured and the INT2_HG_WU bit of the HG_FUNCTIONS_ENABLE register has to be set to 1.

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

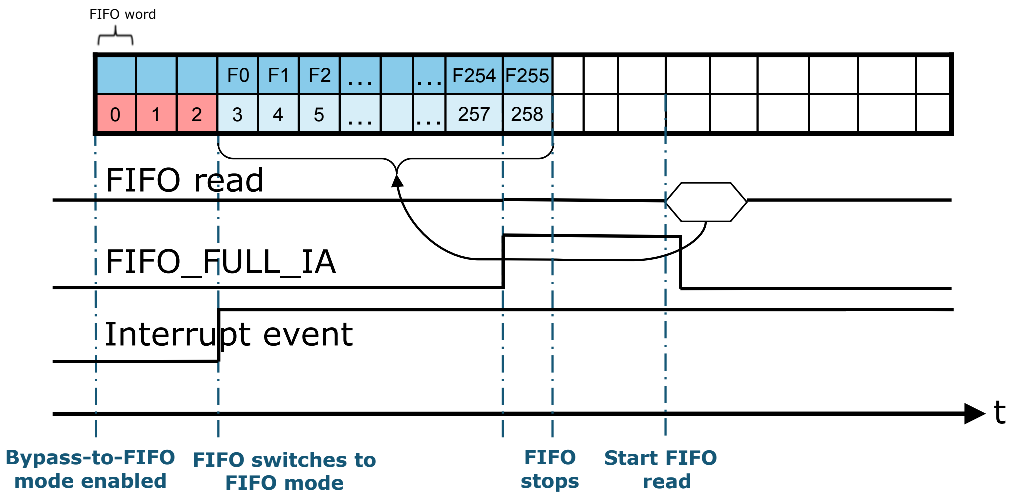
Follow these steps for bypass-to-FIFO mode configuration (if the low-*g* accelerometer/high-*g* accelerometer/gyroscope data-ready is used as the FIFO trigger):

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

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

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

**Figure 24. Bypass-to-FIFO mode**



### 8.7.7 ContinuousWTM-to-full mode

This mode is similar to continuous-to-FIFO mode previously described, with the following additional behaviors:

- When in continuous mode, the FIFO size is automatically limited according to the selected FIFO threshold level (WTM_[7:0] field of the FIFO_CTRL1 register), and for this reason it is referred to as "continuousWTM" mode. When in this mode, the FIFO full event is internally masked.
- When in FIFO mode, the FIFO size is no longer limited to the selected FIFO threshold level, and for this reason it is referred to as "full" mode. When in this mode, the FIFO full event is no longer internally masked.

In continuousWTM-to-full mode, the FIFO buffer starts operating in continuousWTM mode and switches to full mode when an event condition occurs.

The event condition can be one of the following:

- Single tap: event detection has to be configured and the INT2_SINGLE_TAP bit of the MD2_CFG register has to be set to 1.
- Double tap: event detection has to be configured and the INT2_DOUBLE_TAP bit of the MD2_CFG register has to be set to 1.
- Free-fall: event detection has to be configured and the INT2_FF bit of the MD2_CFG register has to be set to 1.
- Wake-up: event detection has to be configured and the INT2_WU bit of the MD2_CFG register has to be set to 1.
- 6D: event detection has to be configured and the INT2_6D bit of the MD2_CFG register has to be set to 1.
- High-*g* wake-up: event detection has to be configured and the INT2_HG_WU bit of the HG_FUNCTIONS_ENABLE register has to be set to 1.

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

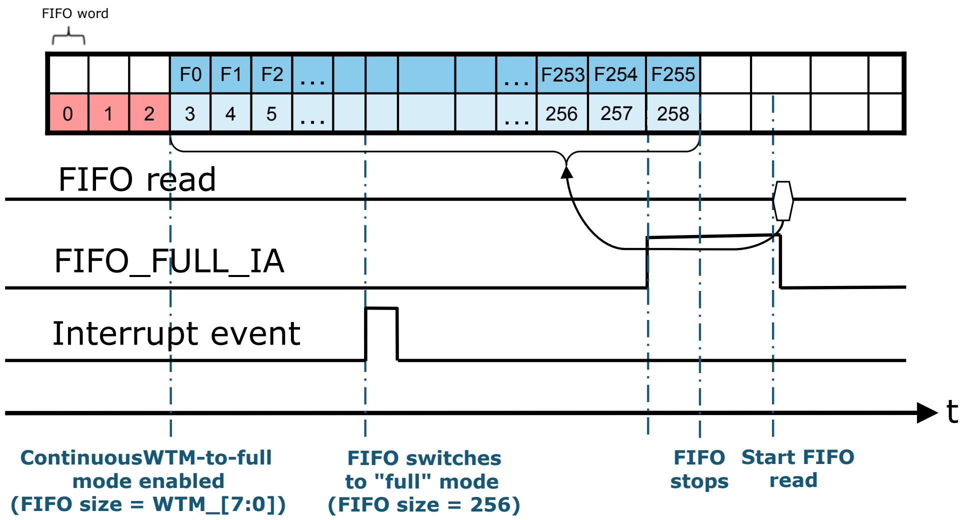
Follow these steps for continuousWTM-to-full mode configuration (if the low-*g* accelerometer/high-*g* accelerometer/gyroscope data-ready is used as the FIFO trigger):

1. Configure one of the events as previously described.
2. Enable the sensor data to be stored in FIFO with the corresponding batch data rate (if configurable).
3. Set the FIFO_MODE_[2:0] field in the FIFO_CTRL4 register to 010 to enable FIFO continuousWTM-to-full mode.

In continuousWTM-to-full mode the FIFO buffer continues filling. When the FIFO is full or overrun at the next FIFO write operation (as indicated above, the FIFO size is automatically limited to the value of the WTM_[7:0] field in the FIFO_CTRL1 register), the FIFO_FULL_IA bit does not go high, since it is internally masked. When the trigger event occurs, the FIFO buffer size is no longer limited to the value of the WTM_[7:0] field in the FIFO_CTRL1 register and it continues filling until it becomes full and then it stops collecting data.

ContinuousWTM-to-full mode can be used in order to analyze the history of both the samples that have generated an interrupt and the samples right after the interrupt generation. The standard operation is to read the FIFO content when the FIFO mode is triggered and the FIFO buffer is full and stopped.

Figure 25. ContinuousWTM-to-full mode



## 8.8 Retrieving data from the FIFO

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

One way to retrieve data from the FIFO is the following:

1. Read the FIFO_STATUS1 and FIFO_STATUS2 registers to check how many words are stored in the FIFO. This information is contained in the DIFF_FIFO_[8:0] field.
2. For each word in FIFO, read the FIFO word (tag and output data) and interpret it on the basis of the FIFO tag.
3. Go to step 1.

The entire FIFO content is retrieved by performing a certain number of read operations from the FIFO output registers until the buffer becomes empty (DIFF_FIFO_[8:0] bits of the FIFO_STATUS1 and FIFO_STATUS2 register are equal to 0).

FIFO can be read when it is empty. In this case, the FIFO word is marked by the specific empty tag.

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

1. Read the FIFO_STATUS1 and FIFO_STATUS2 registers to check how many words are stored in the FIFO. This information is contained in the DIFF_FIFO_[8:0] field.
2. Read DIFF_FIFO + N words with a multiple operation (that is, (DIFF_FIFO + N) * 7 bytes), where N is chosen in order to make sure that the FIFO has been emptied.
3. If the data read from the FIFO do not contain data marked with the empty tag, then read N additional samples in order to empty the FIFO.

## 8.9 FIFO watermark threshold

The FIFO threshold is a functionality of the LSM6DSV80X FIFO that can be used to check when the number of samples in the FIFO reaches a defined watermark threshold level.

The bits WTM_[7:0] in the FIFO_CTRL1 register contain the watermark threshold level. The resolution of the WTM_[7:0] field is 7 bytes, corresponding to a complete FIFO word. So, the user can select the desired level in a range between 0 and 255.

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

**Figure 26. FIFO threshold (STOP_ON_WTM = 0)**

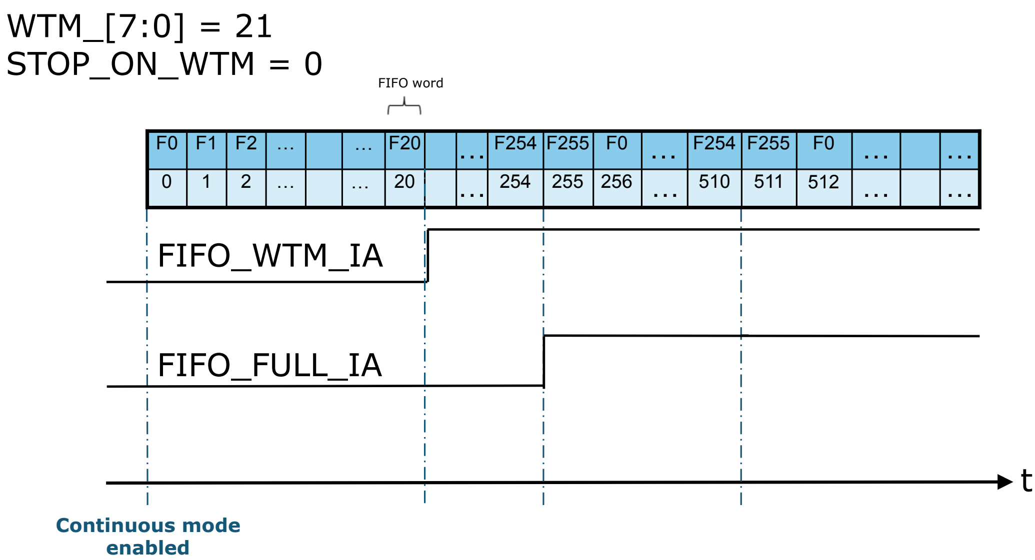



Figure 26. FIFO threshold (STOP_ON_WTM = 0) shows an example of FIFO threshold level usage when just low-*g* accelerometer (or gyroscope) data are stored. The STOP_ON_WTM bit is set to 0 in the FIFO_CTRL2 register. The threshold level is set to 21 through the WTM_[7:0] bits. The FIFO_WTM_IA bit of the FIFO_STATUS2 register rises after the 21$^{st}$ level has been reached (21 words in the FIFO). Since the STOP_ON_WTM bit is set to 0, the FIFO does not stop at the 21$^{st}$ set of data, but keeps storing data until the FIFO_FULL_IA flag is set high.

Figure 27. **FIFO threshold (STOP_ON_WTM = 1) in FIFO mode**

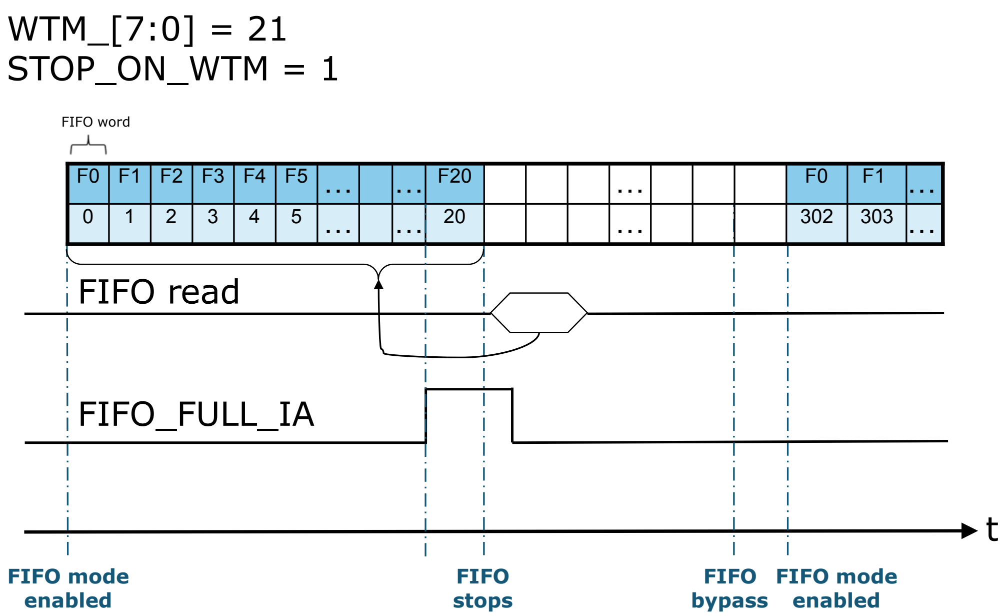



Figure 27. FIFO threshold (STOP_ON_WTM = 1) in FIFO mode shows an example of FIFO threshold level usage in FIFO mode with the STOP_ON_WTM bit set to 1 in the FIFO_CTRL2 register. Just low-*g* accelerometer (or gyroscope) data are stored in this example. The threshold level is set to 21 through the WTM_[7:0] bits and defines the current FIFO size. In FIFO mode, data are stored in the FIFO buffer until the FIFO is full. The FIFO_FULL_IA bit of the FIFO_STATUS2 register rises when the next data stored in the FIFO generates the FIFO full or overrun condition. The FIFO_WTM_IA bit of the FIFO_STATUS2 register goes high when the FIFO is full.

Figure 28. **FIFO threshold (STOP_ON_WTM = 1) in continuous mode**

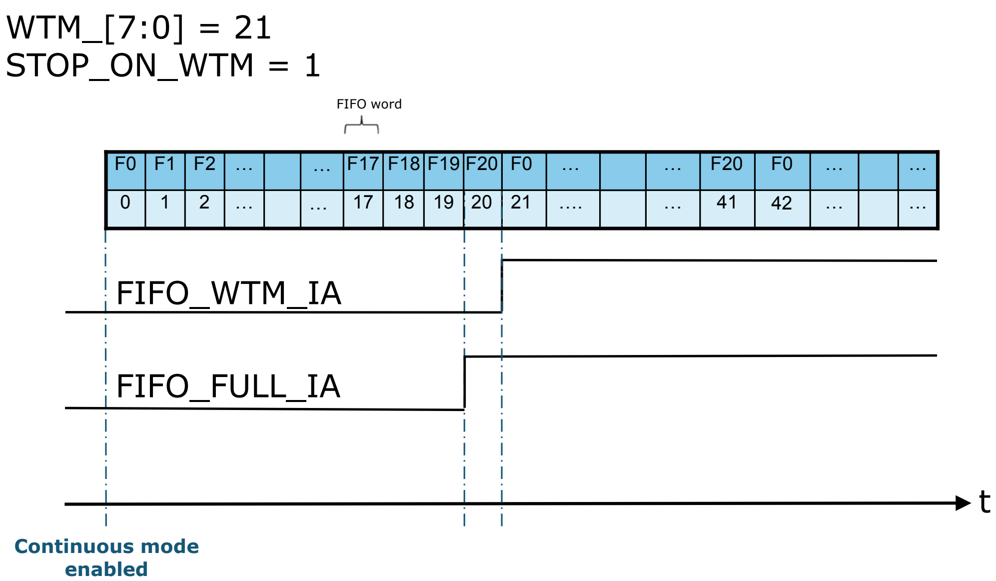


Figure 28. FIFO threshold (STOP_ON_WTM = 1) in continuous mode shows an example of FIFO threshold level usage in continuous mode with the STOP_ON_WTM bit set to 1 in the FIFO_CTRL2 register. Just low-*g* accelerometer (or gyroscope) data are stored in this example. The threshold level is set to 21 through the WTM_[7:0] bits. The FIFO_FULL_IA bit of the FIFO_STATUS2 register rises when the next data stored in the FIFO makes the FIFO full. The FIFO_WTM_IA bit of the FIFO_STATUS2 goes high when the FIFO is full. If data are not retrieved from FIFO, new data (labeled as sample 21) overrides the older data stored in FIFO (labeled as sample F0).

## 8.10 FIFO compression

FIFO compression is an embedded algorithm that allows storing up to three times the number of low-*g* accelerometer and gyroscope data in FIFO. The compression algorithm automatically analyzes the slope of the sensor waveform and applies the compression of data in FIFO on the basis of the slope (difference between two consecutive samples).

FIFO compression can be enabled on low-*g* accelerometer and gyroscope data in FIFO by setting both the FIFO_COMPR_EN bit in the EMB_FUNC_EN_B embedded function register and the FIFO_COMPR_RT_EN bit in the FIFO_CTRL2 register. When active, the compression affects both low-*g* accelerometer and gyroscope data and the level of compression is independent.

The low-*g* accelerometer and gyroscope batch data rate (BDR) can be configured independently, but the compression algorithm is not supported if the low-*g* accelerometer and/or the gyroscope are batched at a rate greater than 1920 Hz.

FIFO compression supports three different levels of compression:

- NC, noncompressed. If the difference between the actual and previous data is higher than 128 LSB, then one sensor sample is stored in one FIFO word.
- 2xC, low compression. If the difference between the actual and previous data is between 16 and 128 LSB, then two sensor samples are stored in one FIFO word.
- 3xC, high compression. If the difference between the actual and previous data is less than 16 LSB, then three sensor samples are stored in one FIFO word.

### 8.10.1 Time correlation

There are five different tags (for each main sensor) depending on the degree of compression:

- NC, noncompressed, associated to the actual time slot
- NC_T_2, noncompressed, associated to two times the previous time slot
- NC_T_1, noncompressed, associated to the previous time slot
- 2xC, low compression
- 3xC, high compression

All NC tags are useful in understanding the time slot correlation. By decoding the sensor tag, it is possible to understand the time frame in which the data was generated.

At the first batch event, the compression algorithm writes a noncompressed word (NC) in FIFO. After that, the algorithm analyzes the slope of the waveforms and three FIFO entries are possible:

- 3xC data written, which contains diff(i), diff(i – 1) and diff(i – 2)
- 2xC data written, which contains diff(i – 1) and diff(i – 2)
- NC_T_2 data written, which contains data(i – 2)

Noncompressed tag sensor NC_T_1 could be written when a configuration change occurs or when the user wants to temporarily disable the runtime FIFO compression by deasserting the FIFO_COMPR_RT_EN bit in the FIFO_CTRL2 register.

The table below summarizes the data and time slot associated for each tag.

**Table 99. FIFO compression tags and associated data**

| Tag sensor | Time slot data |
|---|---|
| NC | data(i) |
| NC_T_1 | data(i - 1) |
| NC_T_2 | data(i - 2) |
| 2xC | diff(i - 2), diff(i - 1) |
| 3xC | diff(i - 2), diff(i - 1), diff(i) |

As shown in Table 99, using FIFO compression introduces a latency of 2 / BDR, since the compression acts on a window of three BDR.

### 8.10.2 Data format

A FIFO word of a compressed data contains the information of its slope with respect to its previous data:

$$data(i) = diff(i) + data(i-1)$$

Thus, the last decoded data, data(i-1) in the formula above, must be saved when performing the decompression task.

The following table summarizes the output data format in FIFO for 2xC compressed data.

**Table 100. 2xC compressed data output data format in FIFO**

| Data | Formula |
|---|---|
| diffx(i – 2) | 8bit_signed(FIFO_DATA_OUT_X_L) |
| diffy(i – 2) | 8bit_signed(FIFO_DATA_OUT_X_H) |
| diffz(i – 2) | 8bit_signed(FIFO_DATA_OUT_Y_L) |
| diffx(i – 1) | 8bit_signed(FIFO_DATA_OUT_Y_H) |
| diffy(i – 1) | 8bit_signed(FIFO_DATA_OUT_Z_L) |
| diffz(i – 1) | 8bit_signed(FIFO_DATA_OUT_Z_H) |

The following table summarizes the output data format in FIFO for 3xC compressed data.

**Table 101. 3xC compressed data output data format in FIFO**

| Data | Formula |
|---|---|
| diffx(i – 2) | 5bit_signed(FIFO_DATA_OUT_X[4:0]) |
| diffy(i – 2) | 5bit_signed(FIFO_DATA_OUT_X[9:5]) |
| diffz(i – 2) | 5bit_signed(FIFO_DATA_OUT_X[14:10]) |
| diffx(i – 1) | 5bit_signed(FIFO_DATA_OUT_Y[4:0]) |
| diffy(i – 1) | 5bit_signed(FIFO_DATA_OUT_Y[9:5]) |
| diffz(i – 1) | 5bit_signed(FIFO_DATA_OUT_Y[14:10]) |
| diffx(i) | 5bit_signed(FIFO_DATA_OUT_Z[4:0]) |
| diffy(i) | 5bit_signed(FIFO_DATA_OUT_Z[9:5]) |
| diffz(i) | 5bit_signed(FIFO_DATA_OUT_Z[14:10]) |

In the table above:

- FIFO_DATA_OUT_X[15:0] = FIFO_DATA_OUT_X_L + FIFO_DATA_OUT_X_H << 8
- FIFO_DATA_OUT_Y[15:0] = FIFO_DATA_OUT_Y_L + FIFO_DATA_OUT_Y_H << 8
- FIFO_DATA_OUT_Z[15:0] = FIFO_DATA_OUT_Z_L + FIFO_DATA_OUT_Z_H << 8

### 8.10.3 Disabling FIFO compression at runtime

The FIFO compression introduces a latency of 2 / BDR in the writing of the sensor in FIFO. Using FIFO compression is not indicated when user want to flush FIFO with low latency.

In case both high latency and low latency can be used, FIFO can be configured in the more convenient way also at runtime.

The FIFO_COMPR_RT_EN bit can be changed at runtime in order to move from an enabled compression algorithm to a disabled compression algorithm (without latency). The switching is managed as a device configuration change. FIFO writes the CFG-Change sensor at the first BDR event after the change. In that case, all data not yet stored are written at the same time slot with tag NC, NC_T_2 or NC_T_1.

The table below shows an example of a runtime disabled compression algorithm. In this case, a main sensor, CFG-Change sensor and timestamp sensor are supposed to be batched in FIFO. FIFO compression is runtime disabled between time instant t(i-1) and time instant t(i). As explained above, all data that are not yet stored are written to the same slot preceded by CFG-Change and timestamp sensors.

**Table 102. Example of disabled runtime compression**

| Time | FIFO_COMPR_RT_EN | Sensor | FIFO_DATA_OUT |
|---|---|---|---|
| ... | 1 | ... | ... |
| t(i-3) | 1 | 3xC | diff(i-5), diff(i-4), diff(i-3) |
| t(i-2) | 1 | - | - |
| t(i-1) | 1 | - | - |
| Async event | 0 | - | - |
| t(i) | 0 | CFG_Change | CFG-change data |
| | | Timestamp | Timestamp data |
| | | NC_T_2 | data(i-2) |
| | | NC_T_1 | data(i-1) |
| | | NC | data(i) |
| t(i+1) | 0 | NC | data(i+1) |
| t(i+2) | 0 | NC | data(i+2) |

### 8.10.4 CFG-Change sensor with FIFO compression enabled

When a change of configuration is applied to the device, the application processor must discriminate the data of previous configurations with the data of the new configuration. For this task, the same approach as the FIFO_COMPR_RT_EN change is applied as shown in the table below. In this case, a main sensor, CFG-Change sensor and timestamp sensor are supposed to be batched in FIFO. A new device configuration is applied between time instant t(i-1) and time instant t(i). As explained, all data that are not yet stored are written to the same slot preceded by the CFG-Change and timestamp sensors. After that, the FIFO compression algorithm restarts to operate as expected.

**Table 103. Example of device configuration change with FIFO compression enabled**

| Time | FIFO_COMPR_RT_EN | Sensor | FIFO_DATA_OUT |
|---|---|---|---|
| ... | 1 | ... | ... |
| t(i-3) | 1 | 3xC | diff(i-5), diff(i-4), diff(i-3) |
| t(i-2) | 1 | - | - |
| t(i-1) | 1 | - | - |
| Async event (CFG-change) | 1 | - | - |
| t(i) | 1 | CFG_Change | CFG-change data |
| | | Timestamp | Timestamp data |
| | | NC_T_2 | data(i-2) |
| | | NC_T_1 | data(i-1) |
| | | NC | data(i) |
| t(i+1) | 1 | - | - |
| t(i+2) | 1 | - | - |
| t(i+3) | 1 | 3xC | diff(i+1), diff(i+2), diff(i+3) |

### 8.10.5 Noncompressed data rate

A compression algorithm can be configured in order to guarantee writing of noncompressed data with a certain periodicity (8, 16, 32 BDR events) through the UNCOMPR_RATE_[1:0] field in FIFO_CTRL2.

The usage of the noncompressed data rate in FIFO can be useful for data reconstruction when there is a possibility of FIFO overrun events: if an overrun occurs and the reference noncompressed data is overwritten, it is not possible to reconstruct the current data until new noncompressed data is written in FIFO. The UNCOMPR_RATE_[1:0] field configures the compression algorithm to write noncompressed data at a specific rate, in order to be sure to have at least one noncompressed data every 8, 16, or 32 samples.

**Table 104. UNCOPTR_RATE configuration**

| UNCOPTR_RATE_[1:0] | NC data write |
|---|---|
| 00 | NC data is not forced |
| 01 | NC data each 8 BDR |
| 10 | NC data each 16 BDR |
| 11 | NC data each 32 BDR |

### 8.10.6 FIFO compression initialization

When FIFO is set in bypass mode, the compression algorithm must be reinitialized by asserting the FIFO_COMPR_INIT bit in the EMB_FUNC_INIT_B embedded functions register.

### 8.10.7 FIFO compression example

The following table provides a basic numerical example of the data that could be read from the FIFO when the compression feature is enabled. In this example, the low-*g* accelerometer sensor only is stored in FIFO and it is configured with a full scale of ±2 *g*.

**Table 105. FIFO compression example**

| | FIFO_DATA_OUT registers | | | | | | | Data analysis | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Time [n/ODR] | TAG_SENSOR_[4:0] | X_L | X_H | Y_L | Y_H | Z_L | Z_H | Compression | Acceleration X [LSB] | Acceleration Y [LSB] | Acceleration Z [LSB] | Latency [n/ODR] |
| 0 | 0x02 | 0x4F | 0x01 | 0x84 | 0x00 | 0x85 | 0x3C | NC | 335 | 132 | 15493 | 0 |
| 3 | 0x06 | 0x61 | 0x01 | 0x96 | 0x00 | 0x86 | 0x40 | NC_T_2 | 353 | 150 | 16518 | 2 |
| 4 | 0x09 | 0x5C | 0x0B | 0x43 | 0x0D | 0x33 | 0xF8 | 3xC | 349 | 144 | 16520 | 2 |
| | | | | | | | | | 352 | 154 | 16523 | 1 |
| | | | | | | | | | 339 | 155 | 16521 | 0 |
| 7 | 0x09 | 0x9E | 0x04 | 0x03 | 0xEC | 0xC2 | 0x03 | 3xC | 337 | 159 | 16522 | 2 |
| | | | | | | | | | 340 | 159 | 16517 | 1 |
| | | | | | | | | | 342 | 157 | 16517 | 0 |
| 10 | 0x08 | 0xFB | 0x0A | 0x15 | 0x0E | 0xEE | 0xF0 | 2xC | 337 | 167 | 16538 | 2 |
| | | | | | | | | | 351 | 149 | 16522 | 1 |
| 12 | 0x09 | 0x80 | 0xD8 | 0x64 | 0x20 | 0x97 | 0x2B | 3xC | 351 | 153 | 16512 | 2 |
| | | | | | | | | | 355 | 156 | 16520 | 1 |
| | | | | | | | | | 346 | 152 | 16530 | 0 |

At the first batch event, the compression algorithm writes a noncompressed word (NC) without latency in FIFO. After that, the algorithm analyzes the slope of the waveforms and three possible FIFO entries are possible: 3xC, 2xC, NC_T_2. Noncompressed words with the NC_T_1 tag are not present in this example since there is no runtime configuration change.

The second sample stored in FIFO is a noncompressed word with a latency of 2 samples (NC_T_2): this FIFO entry contains the entire low-*g* accelerometer data (without any compression).

Then, since the low-*g* accelerometer data slope is low, the compression algorithm starts to compress low-*g* accelerometer data: low-*g* accelerometer data should be reconstructed starting from the latest sample just before the current one (the first compressed data is expressed as the difference from the NC_T_2 data, the second compressed data is expressed as the difference from the first compressed data, and so on).

As shown in the example, the compression algorithm works with a three-level depth buffer: if a 2xC compression level is written in FIFO, only the previous data (latency 1) and two times the previous data (latency 2) are stored in the FIFO word.

From the example, the benefit of FIFO compression is also shown: the samples are written in FIFO at interlaced ODR, thus limiting intervention by the host processor even more than normal FIFO usage.

## 8.11 Timestamp correlation

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

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

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

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

$$timestamp = timestamp\left(i-1\right) + \frac{1}{BDR\_MAX}$$

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

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

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

# 9 Temperature sensor

The device is provided with an internal temperature sensor.

If all the sensors are in power-down mode, the temperature sensor is off.

The maximum output data rate of the temperature sensor is 60 Hz and its value depends on how the low-*g* accelerometer, high-*g* accelerometer, and gyroscope sensors are configured:

- If the low-*g* accelerometer is configured in low-power mode, it is the only active sensor (ODR_G = 0000, and ODR_XL_HG = 000), and its ODR is lower than 60 Hz, the temperature data rate is equal to the configured low-*g* accelerometer ODR.
- The temperature data rate is equal to 60 Hz for any other configuration.

For the temperature sensor, the data-ready signal is represented by the TDA bit of the STATUS_REG register. The signal can be driven to the INT2 pin by setting the INT2_DRDY_TEMP bit of the CTRL4 register to 1.

The temperature data is given by the concatenation of the OUT_TEMP_H and OUT_TEMP_L registers and it is represented as a number of 16 bits in two's complement format with a sensitivity of 256 LSB/°C. The output zero level corresponds to 25°C.

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

## 9.1 Example of temperature data calculation

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

**Table 106. Content of output data registers vs. temperature**

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

# 10 Self-test

The embedded self-test functions allow checking the device functionality without moving it. The self-test limits (minimum and maximum possible values of the self-test difference) are indicated in the datasheet.

## 10.1 Low-*g* accelerometer self-test

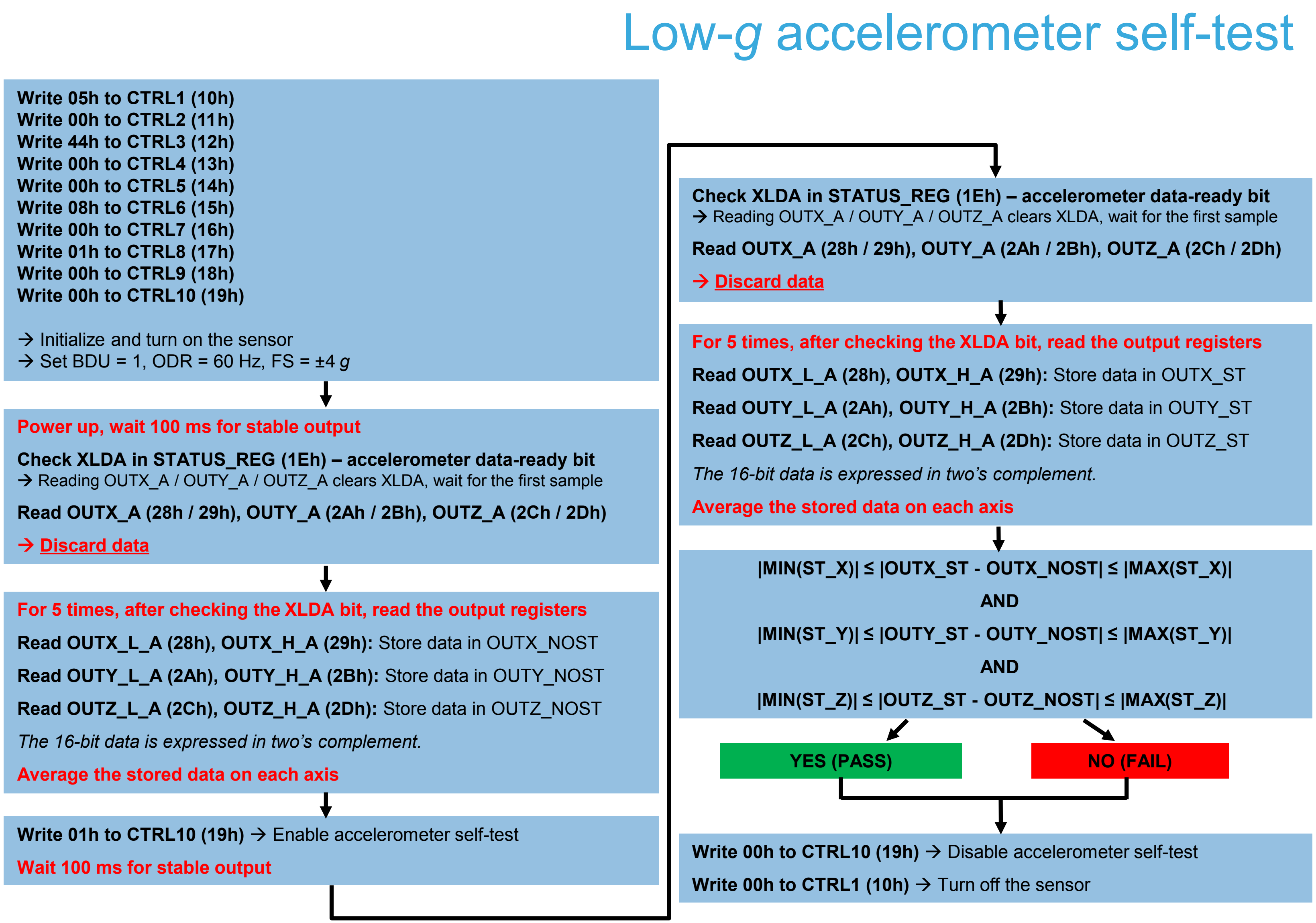
When the low-*g* accelerometer self-test is enabled, an actuation force is applied to the sensor, simulating a definite input acceleration. In this case, the sensor outputs exhibit a change in their DC levels which are related to the selected full scale through the sensitivity value.

The self-test function is off when the ST_XL_[1:0] bits of the CTRL10 register are programmed to 00. It is enabled when the ST_XL_[1:0] bits are set to 01 (positive sign self-test) or 10 (negative sign self-test).

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

The complete accelerometer self-test procedure is indicated in Figure 29. Low-*g* accelerometer self-test procedure.

Figure 29. Low-*g* accelerometer self-test procedure



## 10.2 Gyroscope self-test

The gyroscope self-test allows testing the mechanical and electrical parts of the gyroscope sensor. When it is activated, an equivalent Coriolis signal is emulated at the input of the ASIC front-end and the sensor output exhibits an output change.

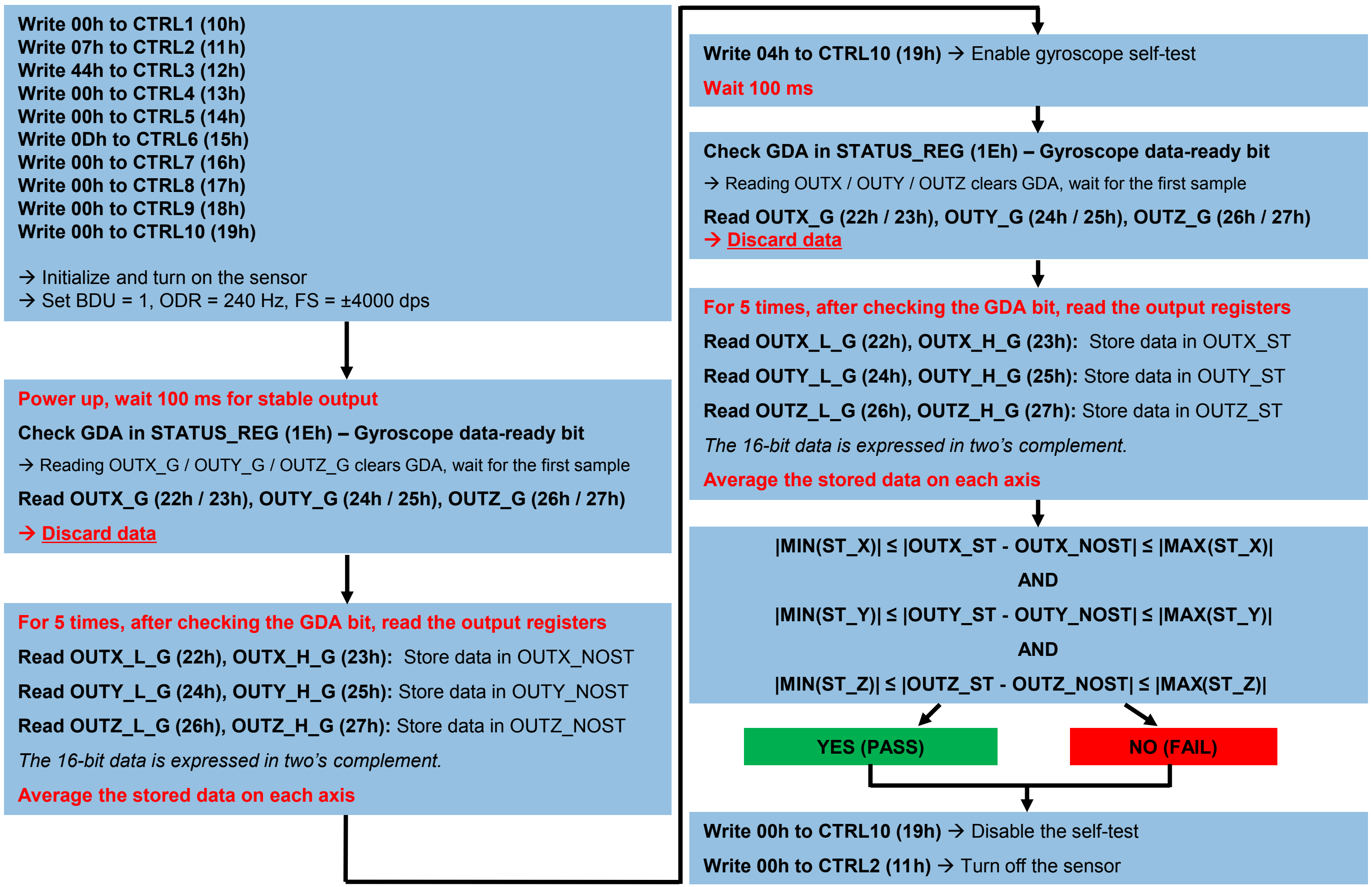
The self-test function is off when the ST_G_[1:0] bits of the CTRL10 register are programmed to 00. It is enabled when the ST_G_[1:0] bits are set to 01 (positive sign self-test) or 10 (negative sign self-test).

When the gyroscope self-test is active, the sensor output level is given by the algebraic sum of the signals produced by the angular rate acting on the sensor and by the electrostatic test-force.

The complete gyroscope self-test procedure is indicated in Figure 30. Gyroscope self-test procedure.

Figure 30. **Gyroscope self-test procedure**

# Gyroscope self-test



## 10.3 High-*g* accelerometer self-test

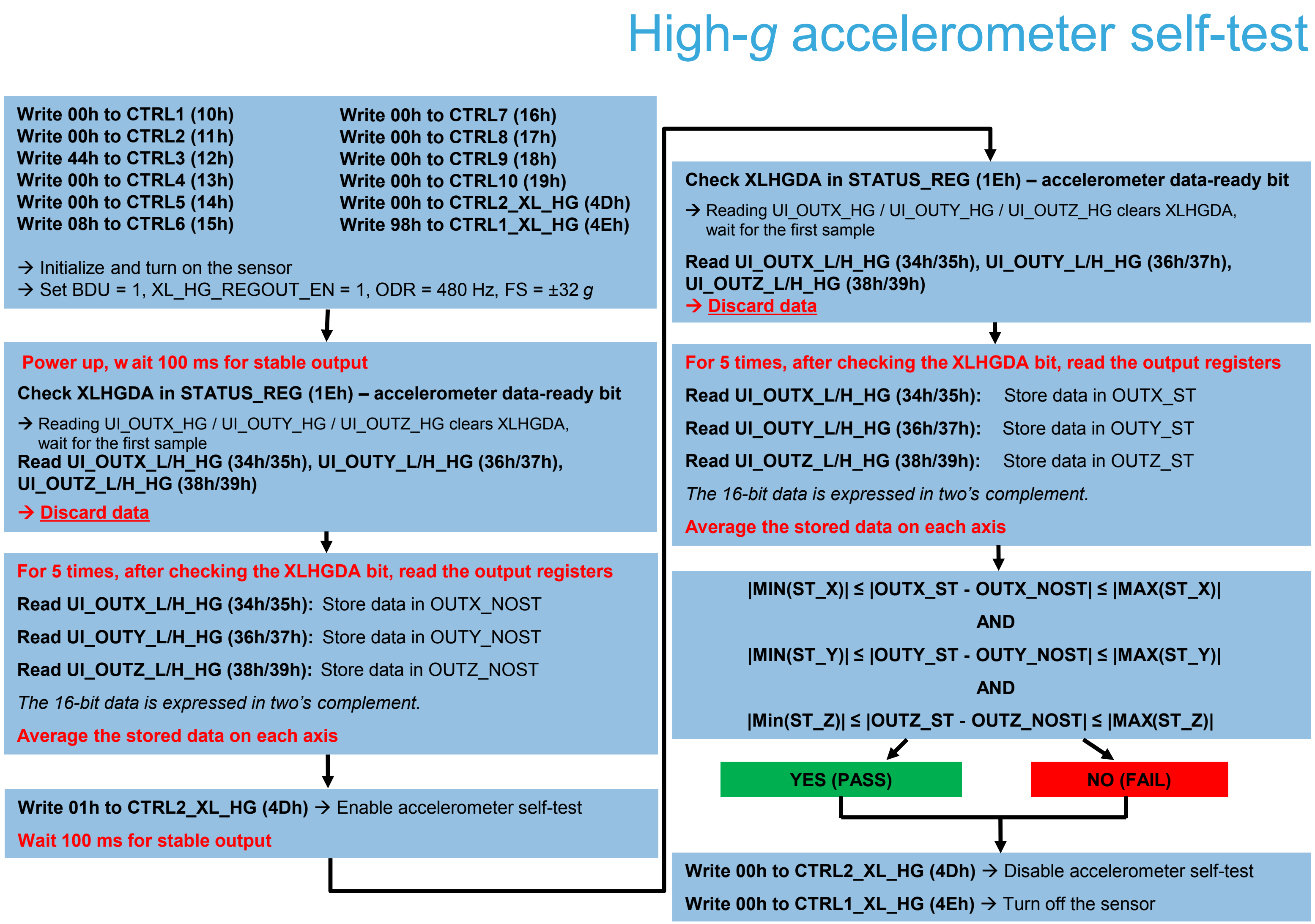
When the high-*g* accelerometer self-test is enabled, an actuation force is applied to the sensor, simulating a definite input acceleration. In this case, the sensor outputs exhibit a change in their DC levels which are related to the selected full scale through the sensitivity value.

The high-*g* accelerometer self-test function is off when the XL_HG_ST[1:0] bits of the CTRL2_XL_HG register are programmed to 00. It is enabled when the XL_HG_ST[1:0] bits are set to 01 (positive sign self-test) or 10 (negative sign self-test).

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

The complete high-*g* accelerometer self-test procedure is indicated in Figure 31. Accelerometer self-test procedure (high-*g*).

Figure 31. Accelerometer self-test procedure (high-*g*)



# Revision history

**Table 107. Document revision history**

| Date | Version | Changes |
| --- | --- | --- |
| 04-Apr-2025 | 1 | Initial release |

# Contents

# List of tables

# List of figures