LIS2DS12：常供电的 3D 加速度计

引言

本文档旨在提供 ST 的 LIS2DS12 运动传感器相关的使用信息和应用提示。

LIS2DS12 是系统级封装的 3D 数字加速度计，具有数字 I^2C/SPI 串口标准输出，在高分辨率模式下功耗 150 µA，在低功耗模式下功耗不超过 80 µA。由于加速度计具有超低噪声性能，始终具有低功耗特性，并结合了高传感精度，因此能够为客户提供最佳运动体验。此外，加速度计具有智能的休眠到唤醒（活动）和返回休眠（不活动）功能，具有先进的节电能力。

该器件具有±2/±4/±8/±16 *g* 的的满量程范围可供客户动态选择，并能通过 1 Hz 到 6400 Hz 的输出数据速率测量加速度。经过配置，LIS2DS12 可利用硬件识别出的自由落体事件、6D 方向、单击和双击感应、活动或不活动、唤醒事件，来生成中断信号。

LIS2DS12 可配置为作为传感器集线器工作。

LIS2DS12 可兼容主要操作系统的要求，提供真实和虚拟传感器。它在硬件中进行了设计，可实现大幅运动检测、倾斜检测和计步功能。

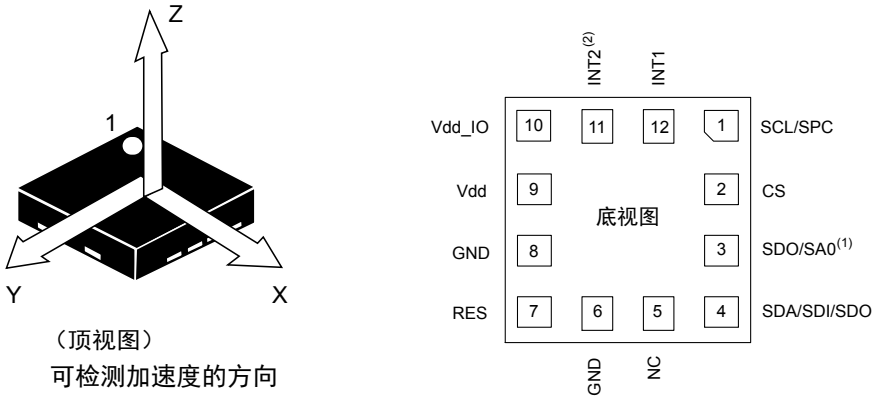
LIS2DS12 集成了 256 级的先进先出（FIFO）缓冲器，允许用户进行数据存储，可减少主机处理器的干预。

LIS2DS12 采用纤薄的小型塑料焊盘栅格阵列封装（LGA），可确保在更大的温度范围（-40 °C 至+85 °C）内正常工作。

SMD 封装的超小尺寸和重量使其成为手持便携式应用的理想选择，如智能手机、物联网（IoT）连接设备，穿戴，以及需要减小封装尺寸和重量的其他应用。

1 引脚说明

图 1. 引脚连接



1. 作为传感器集线器时，该引脚为 I²C 主数据线（MSDA）。
2. 作为传感器集线器时，该引脚为 I²C 主时钟线（MSCL）。

表 1. 引脚说明

引脚#	名称	功能
1	SCL SPC	I²C 串行时钟（SCL） SPI 串口时钟（serial port clock，SPC）
2(1)	CS	SPI 使能 (SPI enable) I²C/SPI 模式选择 （1：SPI 空闲模式/ I²C 通信使能； 0：SPI 通信模式/ I²C 禁用）
3(2)	SDO SA0	SPI 串行数据输出（SDO） 设备地址的 I²C 最低有效位（SA0）
4	SDA SDI SDO	I²C 串行数据（SDA） SPI 串行数据输入（serial data input，SDI） 3 线接口串行数据输出（serial data output，SDO）
5	NC	内部未连接。可连接到 Vdd、Vdd_IO 或 GND。
6	GND	0 V 电源
7	RES	与 GND 连接
8	GND	0 V 电源
9	Vdd	电源
10	Vdd_IO	I/O 引脚的供电
11(3)	INT2	中断引脚 2
12	INT1	中断引脚 1

1. *CS 具有内部上拉，可悬空。*
2. *使用传感器集线器时，该引脚为 I²C 主数据线（MSDA），并在内部设为 0，可通过 FUNC_CTRL (3Fh) 的 TUD_EN 位在内部上拉。*
3. *使用传感器集线器时，该引脚为 I²C 主时钟线（MSCL），可通过 FUNC_CTRL (3Fh) 的 TUD_EN 位在内部上拉。*

2 寄存器

表 2. 寄存器

寄存器名	地址	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
SENSORHUB1_REG	06h	SHub1_7	SHub1_6	SHub1_5	SHub1_4	SHub1_3	SHub1_2	SHub1_1	SHub1_0
SENSORHUB2_REG	07h	SHub2_7	SHub2_6	SHub2_5	SHub2_4	SHub2_3	SHub2_2	SHub2_1	SHub2_0
SENSORHUB3_REG	08h	SHub3_7	SHub3_6	SHub3_5	SHub3_4	SHub3_3	SHub3_2	SHub3_1	SHub3_0
SENSORHUB4_REG	09h	SHub4_7	SHub4_6	SHub4_5	SHub4_4	SHub4_3	SHub4_2	SHub4_1	SHub4_0
SENSORHUB5_REG	0Ah	SHub5_7	SHub5_6	SHub5_5	SHub5_4	SHub5_3	SHub5_2	SHub5_1	SHub5_0
SENSORHUB6_REG	0Bh	SHub6_7	SHub6_6	SHub6_5	SHub6_4	SHub6_3	SHub6_2	SHub6_1	SHub6_0
Module_8bit	0Ch	MODULE_7	MODULE_6	MODULE_5	MODULE_4	MODULE_3	MODULE_2	MODULE_1	MODULE_0
WHO_AM_I	0Fh	0	1	0	0	0	0	1	1
CTRL1	20h	ODR3	ODR2	ODR1	ODR0	FS1	FS0	HF_ODR	BDU
CTRL2	21h	BOOT	SOFT_ RESET	0	FUNC_CFG_E N	FDS_SLOPE	IF_ADD_INC	I2C_ DISABLE	SIM
CTRL3	22h	ST2	ST1	TAP_X_EN	TAP_Y_EN	TAP_Z_EN	LIR	H_LACTIVE	PP_OD
CTRL4	23h	INT1_MASTER _DRDY	INT1_S_TAP	INT1_WU	INT1_FF	INT1_TAP	INT1_6D	INT1_FTH	INT1_DRDY
CTRL5	24h	DRDY _PULSED	INT2_BOOT	INT2_ ON_INT1	INT2_TILT	INT2_ SIG_MOT	INT2_ STEP_DET	INT2_FTH	INT2_DRDY
FIFO_CTRL	25h	FMODE2	FMODE1	FMODE0	INT_STEP_ COUNT_OV	MODULE _TO_FIFO	0	0	IF_CS_PU _DIS
OUT_T	26h	Temp7	Temp6	Temp5	Temp4	Temp3	Temp2	Temp1	Temp0
STATUS	27h	FIFO_THS	WU_IA	SLEEP _STATE	DOUBLE _TAP	SINGLE _TAP	6D_IA	FF_IA	DRDY
OUTX_L	28h	XL_7	XL_6	XL_5	XL_4	XL_3	XL_2	0	0
OUTX_H	29h	XH_7	XH_6	XH_5	XH_4	XH_3	XH_2	XH_1	XH_0
OUTY_L	2Ah	YL_7	YL_6	YL_5	YL_4	YL_3	YL_2	0	0
OUTY_H	2Bh	YH_7	YH_6	YH_5	YH_4	YH_3	YH_2	YH_1	YH_0
OUTZ_L	2Ch	ZL_7	ZL_6	ZL_5	ZL_4	ZL_3	ZL_2	0	0
OUTZ_H	2Dh	ZH_7	ZH_6	ZH_5	ZH_4	ZH_3	ZH_2	ZH_1	ZH_0
FIFO_THS	2Eh	FTH7	FTH6	FTH5	FTH4	FTH3	FTH2	FTH1	FTH0

寄存器名	地址	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
FIFO_SRC	2Fh	FTH	FIFO_OVR	DIFF8	0	0	0	0	0
FIFO_SAMPLES	30h	DIFF7	DIFF6	DIFF5	DIFF4	DIFF3	DIFF2	DIFF1	DIFF0
TAP_6D_THS	31h	4D_EN	6D_THS1	6D_THS0	TAP_THS4	TAP_THS3	TAP_THS2	TAP_THS1	TAP_THS0
INT_DUR	32h	LAT3	LAT2	LAT1	LAT0	QUIET1	QUIET0	SHOCK1	SHOCK0
WAKE_UP_THS	33h	SINGLE_ DOUBLE_ TAP	SLEEP_ON	WU_THS5	WU_THS4	WU_THS3	WU_THS2	WU_THS1	WU_THS0
WAKE_UP_DUR	34h	FF_DUR5	WU_DUR1	WU_DUR0	INT1_FSS7	SLEEP_ DUR3	SLEEP_ DUR2	SLEEP_ DUR1	SLEEP_ DUR0
FREE_FALL	35h	FF_DUR4	FF_DUR3	FF_DUR2	FF_DUR1	FF_DUR0	FF_THS2	FF_THS1	FF_THS0
STATUS_DUP	36h	OVR	WU_IA	SLEEP _STATE	DOUBLE _TAP	SINGLE_ TAP	6D_IA	FF_IA	DRDY
WAKE_UP_SRC	37h	0	0	FF_IA	SLEEP_ STATE_IA	WU_IA	X_WU	Y_WU	Z_WU
TAP_SRC	38h	0	TAP_IA	SINGLE _TAP	DOUBLE _TAP	TAP_SIGN	X_TAP	Y_TAP	Z_TAP
6D_SRC	39h	0	6D_IA	ZH	ZL	YH	YL	XH	XL
STEP_ COUNTER_MINTHS	3Ah	RST_NSTEP	PEDO4g	SC_MTHS5	SC_MTHS4	SC_MTHS3	SC_MTHS2	SC_MTHS1	SC_MTHS0
STEP_ COUNTER_L	3Bh	nSTEP_L7	nSTEP_L6	nSTEP_L5	nSTEP_L4	nSTEP_L3	nSTEP_L2	nSTEP_L1	nSTEP_L0
STEP_ COUNTER_H	3Ch	nSTEP_H7	nSTEP_H6	nSTEP_H5	nSTEP_H4	nSTEP_H3	nSTEP_H2	nSTEP_H1	nSTEP_H0
FUNC_CK_GATE	3Dh	TILT_INT	FS_SRC1	FS_SRC0	SIG_MOT_ DETECT	RST_SIG_ MOT	RST_PEDO	STEP_ DETECT	CK_GATE_ FUNC
FUNC_SRC	3Eh	0	0	0	0	0	RST_TILT	MODULE_ READY	SENSORHUB_ END_OP
FUNC_CTRL	3Fh	0	0	MODULE_ 开启	TILT_ON	TUD_EN	MASTER_ 开启	SIG_MOT_ 开启	STEP_CNT _ON

2.1 高级配置寄存器

高级配置寄存器用于配置特定的器件功能，如计步器和传感器集线器。要切换到高级配置寄存器页面，必须在 CTRL（21h）中将 FUNC_CFG_EN 位置为“1”。要返回标准寄存器页面，必须在寄存器 3Fh 中将 FUNC_CFG_EN 位置为“0”。

注意：所有对高级配置寄存器内容的修改必须在加速度计传感器处于掉电模式时进行。

表 3. 高级配置寄存器

寄存器名	地址	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
PEDO_DEB_REG	2Bh	DEB_TIME4	DEB_TIME3	DEB_TIME2	DEB_TIME1	DEB_TIME0	DEB_STEP2	DEB_STEP1	DEB_STEP0
SLV0_ADD	30h	Slave0_add6	Slave0_add5	Slave0_add4	Slave0_add3	Slave0_add2	Slave0_add1	Slave0_add0	rw_0
SLV0_SUBADD	31h	Slave0_reg7	Slave0_reg6	Slave0_reg5	Slave0_reg4	Slave0_reg3	Slave0_reg2	Slave0_reg1	Slave0_reg0
SLV0_CONFIG	32h	-	-	-	-	-	Slave0_ numop2	Slave0_ numop1	Slave0_ numop0
DATAWRITE_SLV0	33h	Slave0_ dataw7	Slave0_ dataw6	Slave0_ dataw5	Slave0_ dataw4	Slave0_ dataw3	Slave0_ dataw2	Slave0_ dataw1	Slave0_ dataw0
SM_THS	34h	SM_THS_7	SM_THS_6	SM_THS_5	SM_THS_4	SM_THS_3	SM_THS_2	SM_THS_1	SM_THS_0
STEP_COUNT_ DELTA	3Ah	STEP_ COUNT_D7	STEP_ COUNT_D6	STEP_ COUNT_D5	STEP_ COUNT_D4	STEP_ COUNT_D3	STEP_ COUNT_D2	STEP_ COUNT_D1	STEP_ COUNT_D0
CTRL2	3Fh	BOOT(1)	SW_ RESET(1)	0	FUNC_ CFG_EN	FDS_ SLOPE(1)	IF_ADD_ INC(1)	I2C_ DISABLE(1)	SIM(1)

1. *只读位*

3 工作模式

LIS2DS12 器件提供两种功耗模式：高分辨率（HR）/高频模式（HF）和低功耗（LP）模式。
施加电源后，LIS2DS12 执行一段 20 ms 的启动程序来加载修整参数。启动完成后，器件会自动配置为掉电模式。
参考 LIS2DS12 数据手册，可以利用输出数据率（ODR）和 CTRL1 寄存器的高频（HF_ODR）位，来选择功耗模式和加速度计传感器的输出数据率（表 4. 加速度计 ODR 和功耗模式选择）。

表 4. 加速度计 ODR 和功耗模式选择

ODR [3:0]	HF_ODR	模式	ODR 选择[Hz]	分辨率（位数）
0000	0	掉电模式	掉电模式	-
1000	0	LP	1	10
1001	0	LP	12.5	10
1010	0	LP	25	10
1011	0	LP	50	10
1100	0	LP	100	10
1101	0	LP	200	10
1110	0	LP	400	10
1111	0	LP	800	10
0001	0	HR	12.5	14
0010	0	HR	25	14
0011	0	HR	50	14
0100	0	HR	100	14
0101	0	HR	200	14
0110	0	HR	400	14
0111	0	HR	800	14
0101	1	HF	1600	12
0110	1	HF	3200	12
0111	1	HF	6400	12

输出数据具有不同的分辨率，并且是左对齐的。例如，在 10 位分辨率的情况下，输出数据是 OUT_H&OUT_L 级联的 10 个最高有效位，原始值必须右移 6 位。
表 5. 功耗 LIS2DS12 显示了不同工作模式下功耗典型值。

表 5. 功耗

ODR [Hz]	HR/HF（μA）	LP（μA）
1	150	2.5
12.5	150	4
25	150	5.5
50	150	8
100	150	12.5
200	150	22

ODR [Hz]	HR/HF（μA）	LP（μA）
400	150	41
800	150	80
1600	150	-
3200	150	-
6400	150	-

3.1 掉电模式

加速度计处于掉电模式时，器件的全部内部块几乎都会关闭，以最大限度地降低功耗。数字接口（I^2C 和 SPI）仍然在工作，以便能够与器件进行通信。保留配置寄存器的内容而不更新输出数据寄存器，可保持进入掉电模式前存储器中采样的最后数据。

3.2 高分辨率/高频模式

HR/HF 模式下，所有的加速器电路始终接通，并通过 ODR 位来选择生成数据的数据率。数据中断产生是活动的。

HR 模式采用 14 位分辨率，而 HF 模式则采用 12 位分辨率（参见表 4. 加速度计 ODR 和功耗模式选择）。

3.3 低功耗模式

低功耗模式下，加速度计电路周期性地接通/断开，其占空比是所选 ODR 的函数。此模式与 HR/HF 模式的可用输出数据速率不同。低功耗模式下，具有与 HR 模式加 1 Hz 的情形相同的数据速率（从 12.5 Hz 到 800 Hz，但功耗较低）。

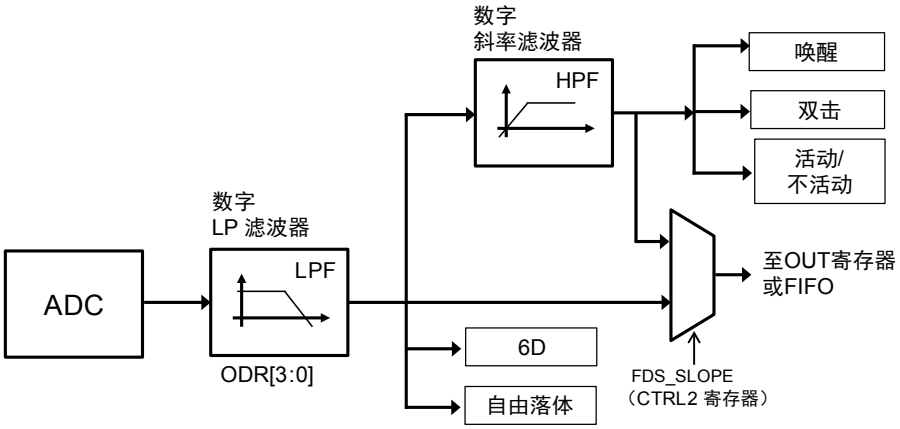
数据中断产生是活动的。

LP 模式采用 10 位分辨率（参见表 4. 加速度计 ODR 和功耗模式选择）。

3.4 加速度计带宽

加速度计采样链（图 2. 加速度计采样链）由几个级联模块表示：ADC 转换器、数字低通滤波器和数字斜率滤波器。

图 2. 加速度计采样链



数字信号由低通数字滤波器（LPF）进行滤波，该滤波器截止频率取决于所选加速度计 ODR，如表 6. 加速度计 LPF1 截止频率中所示。

表 6. 加速度计 LPF1 截止频率

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

发送到 OUT 寄存器的信号（LPF 或 HPF）选择由 CTRL2 寄存器的 FDS_SLOPE 位决定。当它为逻辑“1”时，选择 HPF 信号，否则为 LPF 信号。

发送到数字功能（唤醒，双击，活动/不活动和 6D 方向）的信号始终是 HPF 信号。ADC 采样频率和数字 LPF 截止频率可确保实现抗混叠滤波。抗混叠滤波仅在 HR/HF 模式下可用。当加速度计处于 LP 模式时，电路会周期性地接通/关闭（降低功耗），具有固定的导通时间，并且周期是所选 ODR 的函数。因此，LPF 截止频率固定为 3.2 kHz，所以用户必须注意选择合适的 ODR 值和应用采样频率，以避免混叠（基于所用系统的噪声特性）。

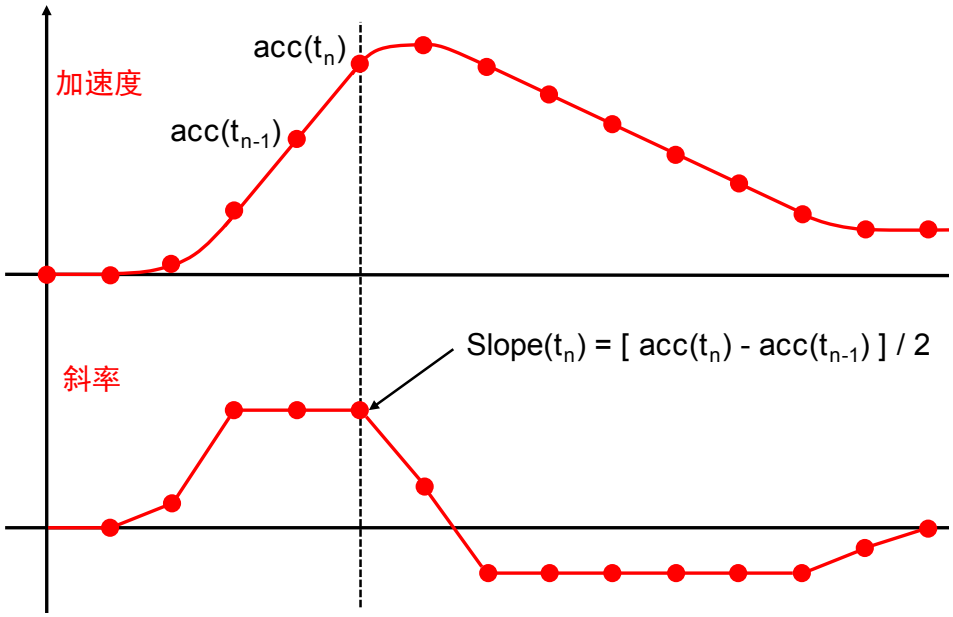
3.4.1 加速度计斜率滤波器

如图 3. 加速度计斜率滤波器中所示，LIS2DS12 器件嵌入了一个数字斜率滤波器，用于唤醒和单击/双击功能。该斜率滤波器输出数据利用以下公式进行计算：

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

图 3. 加速度计斜率滤波器中举例说明了斜率数据信号的示例。

图 3. 加速度计斜率滤波器



斜率滤波器带宽为 ~ ODR/4，通过将 CTRL2（21h）的 FDS_SLOPE 位置为“1”，可在输出寄存器和 FIFO 中获取其数据。

4 读取输出数据

4.1 启动序列

当器件上电时，器件会自动从嵌入的内存中加载校准系数到内部寄存器中。启动程序完成后，也就是大约 20 毫秒后，加速度计会自动进入掉电。

要启用加速度计并采集加速度数据，需要通过 CTRL1 寄存器选择某一种工作模式。

以下通用线序可用来配置加速度计：

1. 写入 CTRL1 = 60h // Acc = 400 Hz（高分辨率模式）
2. 写入 CTRL4 = 01h // INT1 上，Acc 数据准备就绪中断

4.2 使用状态寄存器

该器件提供了一个 STATUS 寄存器，可用于轮询检查新的一组数据什么时候可以用。当加速度计输出中有一组新的数据可用时，DRDY 位被置为 1。

对于加速度计，应当按照如下步骤进行读取：

1. 读 STATUS
2. 如果 DRDY = 0，则进入 1
3. 读 OUTX_L
4. 读 OUTX_H
5. 读 OUTY_L
6. 读 OUTY_H
7. 读 OUTZ_L
8. 读 OUTZ_H
9. 数据处理
10. 调到步骤 1

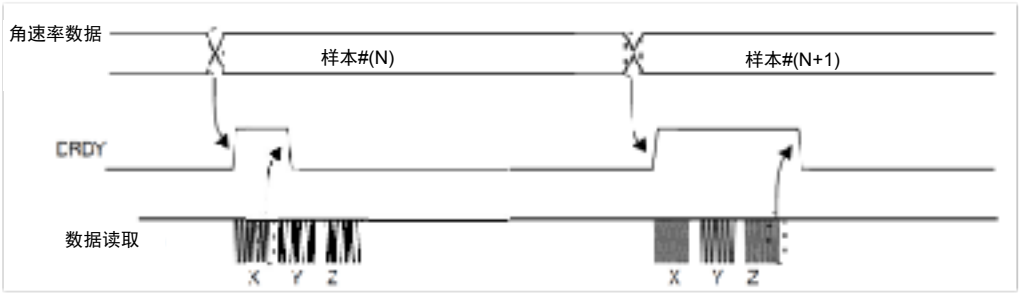
4.3 使用数据准备就绪信号

该器件可配置为具有一个 HW 信号，以确定新的一组测量数据何时可以读取。

对于加速度计传感器，数据准备就绪信号由 STATUS 寄存器的 DRDY 位表示。通过将 CTRL4 寄存器的 INT1_DRDY 位置为 1，可将该信号驱动至 INT1 引脚，通过将 CTRL5 寄存器的 INT2_DRDY 位置为 1，将其驱动至 INT2 引脚。

当一组新数据生成并可读取时，数据准备就绪信号升高为 1。在 DRDY 锁存模式（CTRL5 寄存器中的 DRDY_PULSED 位= 0）下，这是默认条件，当其中一个通道的较高部分已被读取（29h、2Bh、2Dh）时，信号将复位。在 DRDY 脉冲模式（DRDY_PULSED = 1）下，脉冲持续时间约为 75 µs。

图 4. 数据准备就绪信号



4.4 使用块数据更新（block data update，BDU）功能

如果读取加速度计数据特别慢，并且不能（或者不需要）与 STATUS 寄存器中的 DRDY 事件位或驱动到 INT1/INT2 引脚的 DRDY 信号同步，那么强烈建议将 CTRL1（20h）寄存器中的 BDU（块数据更新）位置为 1。

此功能可以避免读取不同样本相关的值（输出数据的最高有效部分和最低有效部分）。特别是在 BDU 被激活的情况下，与每条通道相关联的数据寄存器始终会包含由器件生成的最新输出数据，但如果发起了对数据寄存器（即 OUTX_H 和 OUTX_L、OUTY_H 和 OUTY_L、OUTZ_H 和 OUTZ_L）的读取，读取数据的 MSB 和 LSB 部分之前，都会禁止刷新该寄存器对。

请注意：BDU 只能确保 LSB 部分和 MSB 部分同一时刻被采样。例如，如果读取速度非常慢，则 X 和 Y 可在 T1 读取，Z 在 T2 采样。

4.5 认识输出数据

测得的加速度数据会发送至 OUTX_H、OUTX_L、OUTY_H、OUTY_L、OUTZ_H 和 OUTZ_L 寄存器。这些寄存器分别包含作用于 X、Y 和 Z 轴的加速度信号的最高有效部分和最低有效部分。

X、Y、Z 通道的完整加速度数据是由 OUTX_H & OUTX_L、OUTY_H & OUTY_L、OUTZ_H & OUTZ_L 共同提供的，表示为 2 的补码。

加速度数据表示为 16 位数字，称为 LSB，但根据所选择的工作模式（LP/HR/HF）不同，其分辨率也不同。请参见表 4. 加速度计 ODR 和功耗模式选择。

计算 LSB 后，必须乘以适当的灵敏度参数，得到按 mg 计的相应值。

4.5.1 输出数据示例

以下是一个简单的示例，说明如何使用 LSB 数据并将其转换成 mg。
这些值是在理想器件校准的假设下给出的（即，无偏移，无增益误差，……）。
从传感器获取原始数据（HR 模式，ODR 200 Hz）：

```
OUTX_L: 5Ch
```

```
OUTX_H: FDh
```

```
OUTY_L: 74h
```

```
OUTY_H: 00h
```

```
OUTZ_L: F8h
```

```
OUTZ_H: 42h
```

将寄存器串联：

```
OUTX_H & OUTX_L: FD5Ch
```

```
OUTY_H & OUTY_L: 0074h
```

```
OUTZ_H & OUTZ_L: 42F8h
```

应用灵敏度（例如，满刻度为±2 *g* 时，使用 0.061）：

```
X: -676 * 0.061 = -41 mg
```

```
Y: +116 * 0.061 = +7 mg
```

```
Z: +17144 * 0.061 = +1046 mg
```

5 中断生成和嵌入功能

在 LIS2DS12 器件中，中断生成基于加速度计数据，因此，为了生成中断，加速度计传感器必须设置为活动工作模式（不是掉电）。

可对中断发生器进行配置，来检测：

- 自由落体；
- 唤醒；
- 6D/4D 方向检测；
- 单击和双击感测；
- 活动/不活动检测。

此外，LIS2DS12 能够高效运行先进操作系统中特定的传感器相关功能，节能并且反应时间更快。特别地，经过专门设计，它可在硬件中实现：

- 大幅运动检测；
- 倾斜度检测；
- 计步功能；

所有这些中断信号，以及 FIFO 中断信号和传感器数据就绪信号，可被独立地驱动至 INT1 和 INT2 中断引脚，或通过读取特定源寄存器位分别对其进行检测。

注意：当 MODULE_ON 位置为 1 时，嵌入功能（计步、倾斜和大幅运动检测）不可用。

当 DRDY 信号被发送到中断引脚时，必须使用 CTRL3 寄存器的 H_LACTIVE 位来选择它们的极性。如果该位置为 0（默认值），则中断引脚为高电平激活，当检测到相关中断条件时，这些引脚从低电平变为高电平。否则，如果 H_LACTIVE 位置为 1（低电平激活），则中断引脚正常为高电平，当达到中断条件时，从高电平变为低电平。

当 DRDY 信号被发送到中断引脚时，CTRL3 的 PP_OD 位还允许将这些中断引脚的性质从推挽更改为开漏。如果 PP_OD 位置为 0，则中断引脚处于推挽配置（对于高电平和低电平均为低阻抗输出）。当 PP_OD 位置为 1 时，只有中断活动状态是低阻抗输出。

CTRL3 的 LIR 位可支持中断信号应用锁存模式（不影响 DRDY 信号）。LIR 位置为 1 时，中断引脚一旦被声明，就必须通过读取相关中断源寄存器才能将其复位。如果 LIR 位置为 0，则当不再检测到中断条件或一定时间后（针对不同的中断类型），中断信号可自动复位。

5.1 中断引脚配置

该器件具有两个引脚，可激活这些引脚来生成数据准备就绪或中断信号。这些引脚的功能，对于 INT1 引脚是通过 CTRL4 寄存器来进行选择，对于 INT2 引脚是通过 CTRL5 寄存器来进行选择。

以下是这些中断控制寄存器的说明：这些比特的默认值等于 0，对应于“禁用”。要使能引脚上特定中断信号的线路，须将有关位置为 1。

表 7. **CTRL4 寄存器**

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

- INT1_MASTER_DRDY：管理 INT1 引脚上的主 DRDY 信号
- INT1_S_TAP：单击事件识别被发送到 INT1 引脚。
- INT1_WU：唤醒事件识别被发送到 INT1 引脚。
- INT1_FF：自由落体事件识别被发送到 INT1 引脚。
- INT1_TAP：点击事件识别被发送到 INT1 引脚。
- INT1_6D：6D 事件识别被发送到 INT1 引脚。
- INT1_FTH：FIFO 阈值事件被发送到 INT1 引脚。
- INT1_DRDY：加速度计数据就绪信号被发送到 INT1 引脚。

表 8. **CTRL5 寄存器**

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

- DRDY_PULSED：数据就绪中断模式选择：锁存模式/脉冲模式。
- INT2_BOOT：启动状态发送到 INT2 引脚。
- INT2_ON_INT1：所有 INT2 信号也都发送到 INT1 引脚。
- INT2_TILT：倾斜事件识别发送到 INT2 引脚。
- INT2_SIG_MOT：大幅运动事件识别发送到 INT2 引脚。
- INT2_STEP_DET：计步事件识别发送到 INT2 引脚。
- INT2_FTH：FIFO 阈值事件发送到 INT2 引脚。
- INT2_DRDY：加速度计数据就绪信号在 INT2 引脚。

5.2 事件状态

如果多个中断信号发送到同一个引脚上（INTx），则该引脚的逻辑电平位所选中断信号组合的“或”。为了知道哪个事件产生了中断条件，应用程序要读取正确的状态寄存器，这也会清除事件。

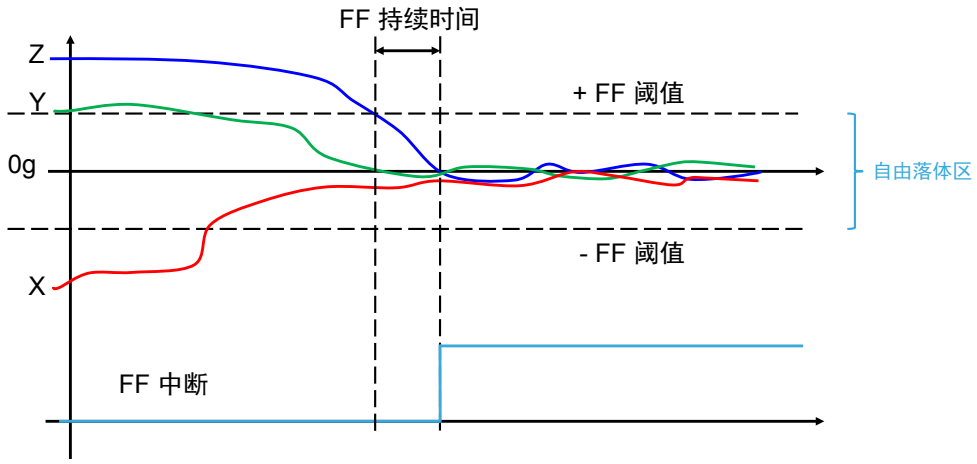
如下所示，STATUS 寄存器在地址 36h 处被复制，以便对连续寄存器（36h/37h/38h/39h）进行多次读取。

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

5.3 自由落体中断

自由落体检测涉及特定的寄存器配置，可以识别器件何时处于自由落体：沿各轴所测得的加速度均为 0。真实情境下，一个“自由落体区域”定义为大约零-g 水平，其中所有加速度均足够小，可以产生中断。自由落体事件检测关联了可配置的阈值和持续时间参数：阈值参数定义了自由落体区幅度；持续时间参数定义了可识别的自由落体中断事件的最小持续时间 （图 5. 自由落体中断）。

图 5. 自由落体中断



通过将 CTRL4 寄存器的 INT1_FF 位置为 1，可将自由落体事件信号传送至 INT1 引脚上；还可通过读取 WAKE_UP_SRC 寄存器的 FF_IA 位对其进行检查。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），则当检测不到自由落体条件时，中断信号会自动复位。如果锁存模式使能且自由落体中断信号被驱动至中断引脚，那么当发生自由落体事件且声明了中断引脚时，必须通过读取 WAKE_UP_SRC 寄存器来将其复位。如果使能了锁存模式，但是中断信号未驱动至中断引脚，那么锁存功能不起作用（当不再核验到自由落体条件时，WAKE_UP_SRC 中的 FF_IA 位复位）。

用来配置阈值参数的寄存器命名为 FREE_FALL；无符号阈值与 FF_THS[2:0]字段值的值相关，如表 9. 自由落体阈值 LSB 值所示，以 31.25 *mg* 为单位表示。此表中给出的 LSB 值对于任何加速度计满量程值均有效。

表 9. 自由落体阈值 LSB 值

FREE_FALL - FF_THS[2:0]	阈值 LSB 值
000	5
001	7
010	8
011	10
100	11

FREE_FALL - FF_THS[2:0]	阈值 LSB 值
101	13
110	15
111	16

持续时间在 N/ODR 中测得，其中 N 为 FREE_FALL / WAKE_UP_DUR 寄存器 FF_DUR[5:0]字段的内容，ODR 为加速度计数据率。

下面给出了自由落体事件识别的基本 SW 程序。

1.	将 60h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz, FS = ±2 *g*
2.	将 00h 写入 WAKE_UP_DUR	// 设置事件持续时间（FF_DUR5 = 0）
3.	将 33h 写入 FREE_FALL	// 设置 FF 阈值（FF_THS[2:0] = 011b） // 设置六个采样事件持续时间（FF_DUR[5:0] = 000110b）
4.	将 10h 写入 CTRL4	// FF 中断驱动至 INT1 引脚
5.	将 04h 写入 CTRL3	// 锁存中断

示例代码中利用设置为~310 mg（31.25 mg * 10）的阈值，用于自由落体识别，该事件由硬件通过 INT1 引脚进行通知。FREE_FALL / WAKE_UP_DUR 寄存器的 FF_DUR[5:0]字段像这样配置：忽略短于 6/ODR = 6/400 Hz = 15 ms 的事件，以避免错误检测。

5.4 唤醒中断

在 LIS2DS12 器件中，唤醒功能可利用斜率滤波器（更多详细信息参见 Section 3.4.1 加速度计斜率滤波器）来实现，如图 3. 加速度计斜率滤波器中所示。如果一定数量的连续斜率滤波数据超出了所配置阈值（图 6. 唤醒中断），则产生唤醒中断信号。

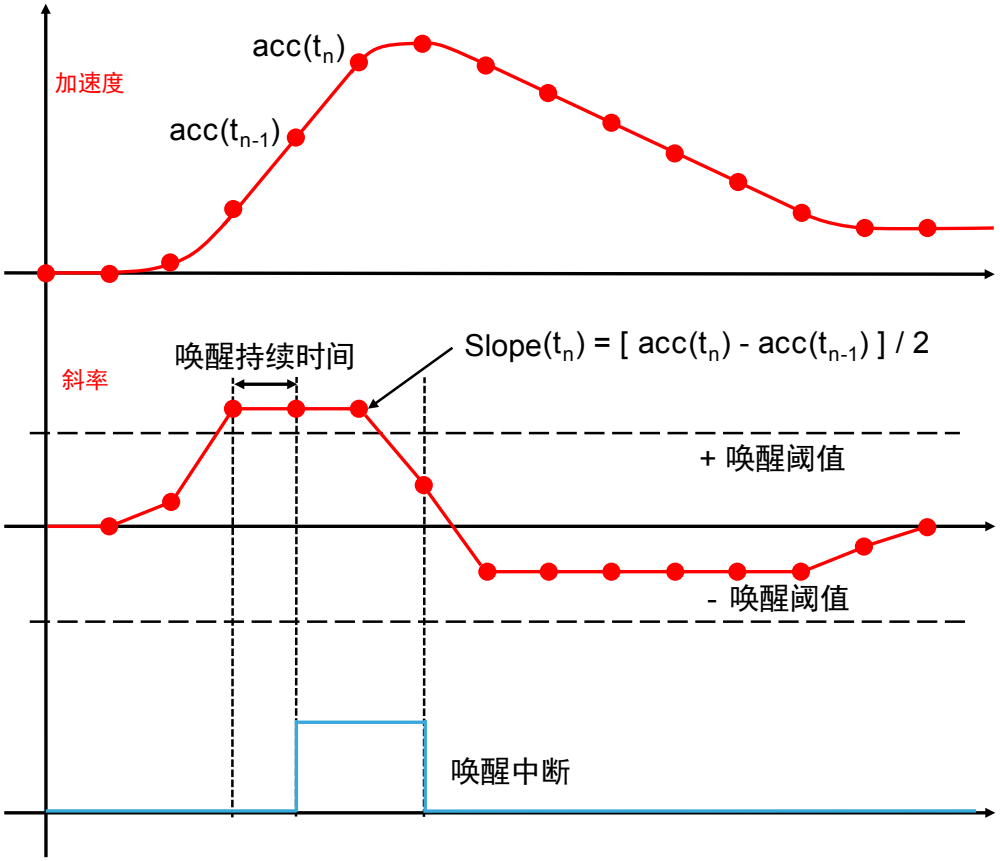
该无符号阈值由 WAKE_UP_THS 寄存器的 WU_THS[5:0]位来定义；这些 6 位的 1 LSB 值取决于所选加速度计满量程：1 LSB = FS/64。阈值可应用于正负数据：对于唤醒中断生成，三个轴中至少有一个必须大于阈值。

持续时间参数定义了所识别的唤醒事件的最小持续时间；其值由 WAKE_UP_DUR 寄存器的 WU_DUR[1:0]位来设置：1 LSB 对应于 1*ODR 时间，这里 ODR 为加速度计输出数据率。要避免因输入信号寄生尖峰而产生不期望的唤醒中断，适当定义持续时间参数是非常重要的。

通过将 CTRL4 寄存器的 INT1_WU 位置为 1，可将该中断信号驱动至 INT1 中断引脚上；还可通过读取 WAKE_UP_SRC 寄存器的 WU_IA 位对其进行检查。WAKE_UP_SRC 寄存器的 X_WU、Y_WU、Z_WU 位指示哪个轴触发了唤醒事件。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），则当滤波数据低于阈值时，中断信号会自动复位。如果锁存模式使能且唤醒中断信号被驱动至中断引脚，那么当发生唤醒事件且声明了中断引脚时，必须通过读取 WAKE_UP_SRC 寄存器来将其复位。如果使能了锁存模式，但是中断信号未驱动至中断引脚，那么锁存功能不起作用（当不再检验到自由落体条件时，WAKE_UP_SRC 中的 WU_IA 位复位）。

图 6. 唤醒中断



下面给出的示例代码实现了用于唤醒事件识别的 SW 程序。

1.	将 60h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz, FS = ±2 *g*
2.	将 00h 写入 WAKE_UP_DUR	// 无持续
3.	将 02h 写入 WAKE_UP_THS	// 设置唤醒阈值
4.	将 20h 写入 CTRL4	// 唤醒中断驱动至 INT1 引脚

由于持续时间被置为 0，因此每个 X、Y、Z 滤波数据超出所配置阈值时，会生成唤醒中断信号。WAKE_UP_THS 寄存器的 WU_THS 字段被置为 000010b，因此唤醒阈值为 62.5 mg（= 2 * FS / 64）。

5.5 6D/4D 定向检测

LIS2DS12 器件能够检测空间中器件的方向，可以很容易地实现移动设备的节能程序和自动图像旋转。

5.5.1 6D 定向检测

可以检测器件在空间中的六个方向；当器件从一个方向转向另一个方向时，中断信号被声明。只要保持其位置，中断就不会重新声明。

当只有一个轴超出所选阈值，其他两轴上测得的加速度值低于阈值时，会产生 6D 中断：6D_SRC 寄存器的 ZH、ZL、YH、YL、XH、XL 位可表示出哪个轴触发了 6D 事件。

更具体地说：

表 10. 6D_SRC 寄存器

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

- 当器件从一个方向转向另一个方向时，6D_IA 被置为高电平。
- 当垂直于 Z（Y、X）轴的面几乎是平面，Z（Y、X）轴上测得的加速度为正且模块中的加速度大于阈值时，ZH（YH、XH）被置为高电平。
- 当垂直于 Z（Y、X）轴的面几乎是平面，Z（Y、X）轴上测得的加速度为负且模块中的加速度大于阈值时，ZL（YL、XL）被置为高电平。

TAP_6D_THS 寄存器的 6D_THS[1:0]位用来选择阈值，该阈值用于检测器件方向变化。表 11. 4D/6D 功能阈值中给出的阈值对于每种加速度计满量程值均有效。

表 11. 4D/6D 功能阈值

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

通过将 CTRL4 寄存器的 INT1_6D 位置为 1，可将该中断信号驱动至 INT1 中断引脚上；还可通过读取 6D_SRC 寄存器的 6D_IA 位对其进行检查。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），则中断信号仅激活 1/ODR[s]，然后自动失效（ODR 为加速度计输出数据率）。如果锁存模式使能，并且 6D 中断信号被驱动至中断引脚，那么当方向发生了改变且中断引脚被声明时，对 6D_SRC 寄存器的读取会清除请求，器件将识别另一个不同的方向。如果使能了锁存模式，但是中断信号未驱动至中断引脚，那么锁存功能不起作用。

参考图 7. 6D 识别方向中所示的六种可能情形，表 12. 用于 6D 定位的 6D_SRC 寄存器中显示了每个位置对应的 6D_SRC 寄存器内容所示。

图 7. 6D 识别方向

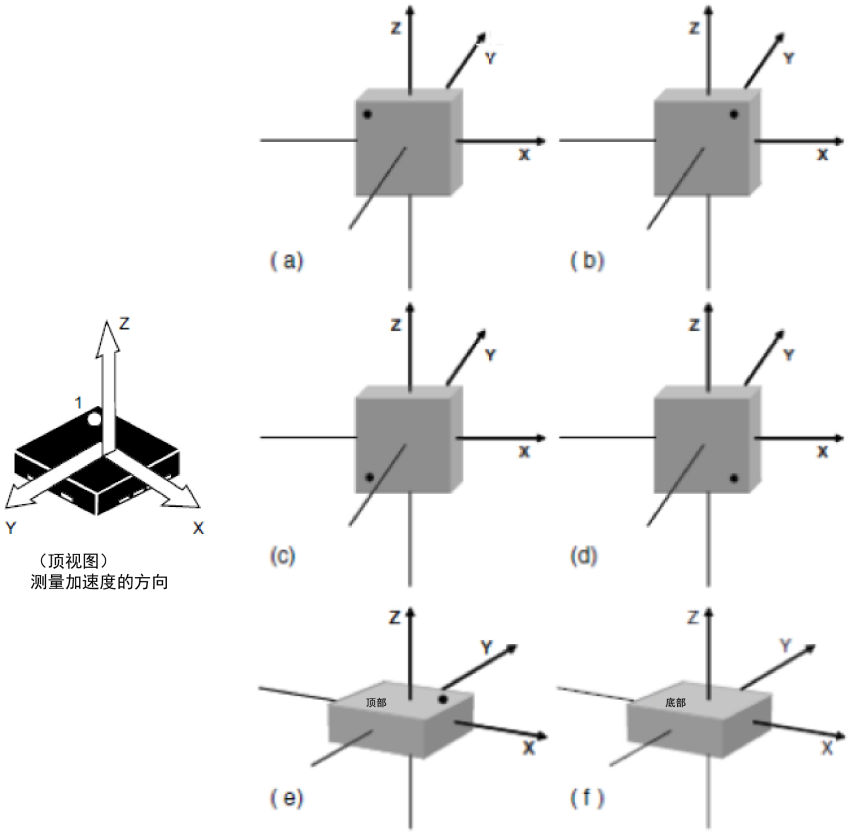


表 12. 用于 6D 定位的 6D_SRC 寄存器

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

下面的示例实现了用于 6D 方向检测的 SW 程序：

1. 将 60h 写入 CTRL1 // 启动加速度计 // ODR = 400 Hz, FS = ±2 *g*
2. 将 40h 写入 TAP_6D_THS // 设置 6D 阈值（6D_THS[1:0] = 10b = 60 degrees）
3. 将 04h 写入 CTRL4 // 6D 中断驱动至 INT1 引脚

5.5.2 4D 方向检测

4D 方向功能是 6D 功能的子集，它被专门定义来进行移动设备中的纵向和横向计算。它可通过将 TAP_6D_THS 寄存器的 4D_EN 位置为 1 来使能。这种配置下，Z 轴位置检测被禁用，因此位置识别减少为表 12. 用于 6D 定位的 6D_SRC 寄存器的(a)、(b)、(c)和(d)的情形。

5.6 单击和双击识别

LIS2DS12 具有单击和双击识别功能，能够在极少加载软件的情况下帮助创建人机界面。器件可配置为沿任意方向点击时在专用引脚上输出中断信号。

如果传感器施加单个输入激励，那么它会在中断引脚 INT1 上产生中断请求。更先进的功能可在识别到两次输入刺激（两个事件的间隔时间可通过程序设定）时生成中断请求，从而可实现类似鼠标按键的功能。

LIS2DS12 器件中，单击和双击识别功能利用两个连续加速度采样之间的斜率来检测点击事件；斜率数据利用以下公式计算：

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

此功能可完全由用户编程，利用专门的寄存器组对所期望的斜率数据幅度和时序进行编程。

单击和双击识别仅在 ODR ≥ 400 Hz 时有意义。

5.6.1 单击

如果器件配置为单击事件检测，那么当所选通道的斜率数据超出了所编程阈值时，会产生一个中断，并在 Shock 时间窗口内返回低电平。

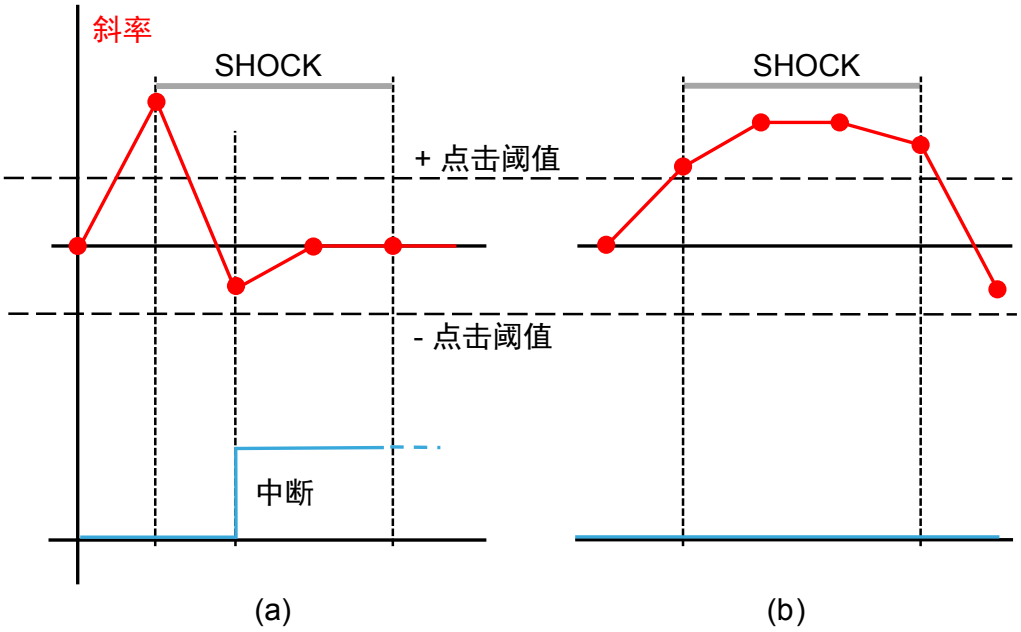
在单击情况下，如果 CTRL3 寄存器的 LIR 位被置为 0，则中断在 Quiet 窗口持续时间内保持高电平。

为了在单击中断信号上使能锁存功能，CTRL3 的 LIR 位必须置为 1：中断保持高电平，直至 TAP_SRC 寄存器被读取。

要实现仅使能单击识别，则 WAKE_UP_THS 的 SINGLE_DOUBLE_TAP 位必须置为 0。

图 8. 单击事件识别的情况（a）中识别出了单击事件，而在情况（b）中，由于在经过 Shock 时间窗口之后斜率数据低于阈值，因此未识别出点击。

图 8. 单击事件识别



5.6.2 双击

如果器件配置为双击事件检测，那么在第一次点击后、识别出第二次点击时，生成中断。只有当事件满足 Shock、Latency 和 Quiet 时间窗口所定义的规则时，才进行第二次点击识别。

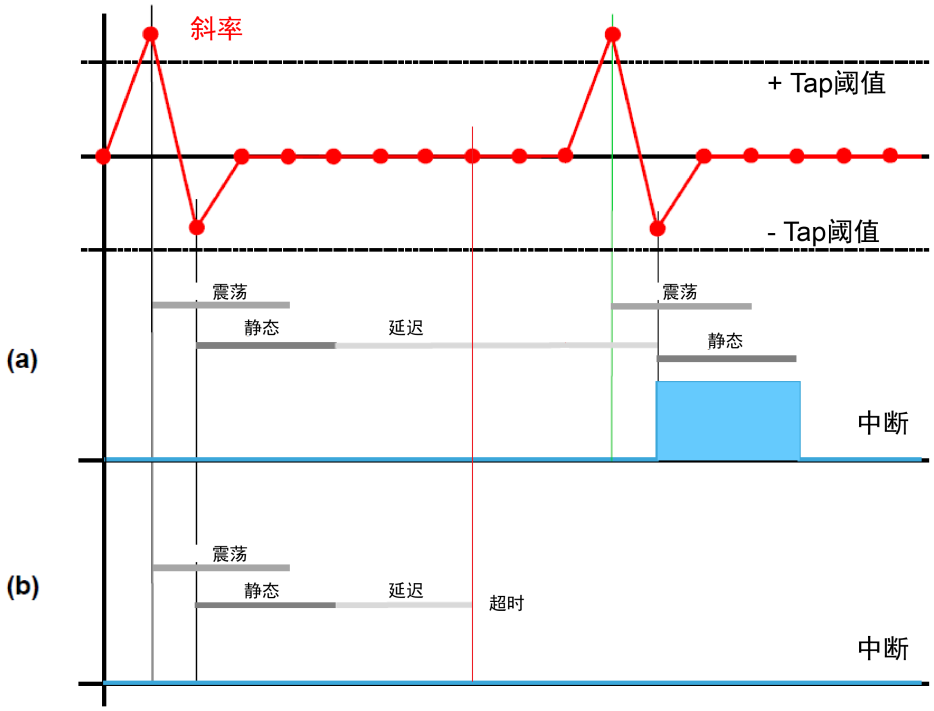
特别地，识别出第一次点击后，第二次点击检测过程会延迟 Quiet 时间所定义的时间间隔。这意味着，识别出第一次点击后，只有在 Quiet 窗口之后、Latency 窗口结束前，斜率数据超过阈值时，才开始第二次点击识别过程。图 9. 双击事件识别（LIR 位 = 0）的情况（a）中，正确识别出了双击事件，而在情况（b）中，由于在经过 Latency 窗口间隔之后斜率数据超出了阈值，因此未产生中断。

一旦第二次点击检测过程开始，则会按照与第一次相同的规则来识别第二次点击：在 Shock 窗口结束之前，斜率数据必须返回到低于阈值之下。

要避免因输入信号伪突变而产生不期望的点击，适当定义 Quiet 窗口是非常重要的。

在双击情况下，如果 CTRL3 寄存器的 LIR 位被置为 0，则中断在 Quiet 窗口持续时间内保持高电平。如果 LIR 位被置为 1，则中断保持高电平直至 TAP_SRC 寄存器被读取。

图 9. 双击事件识别（LIR 位 = 0）



5.6.3 单击和双击识别配置

可对 LIS2DS12 器件进行配置，使其在任一方向发生点击（一次或两次）时均输出中断信号：CTRL3 寄存器的 TAP_X_EN、TAP_Y_EN 和 TAP_Z_EN 位必须置为 1，分别使能 X、Y、Z 方向上的点击识别。

点击识别功能的可配置参数为点击阈值和 shock、quiet 和 latency 时间窗。有效的 ODR 为 400 Hz、800 Hz 和 1600 Hz。

TAP_6D_THS 寄存器的 TAP_THS[4:0]位用来选择用于检测点击事件的无符号阈值。这 5 个比特的 1 LSB 值取决于所选加速度计满量程：1 LSB = FS/32。无符号阈值可应用于正负斜率数据上。

Shock 时间窗口定义了超阈值事件的最大持续时间：在 Shock 窗口结束前，加速度必须返回到低于阈值之下，否则不能检测到该点击事件。INT_DUR 寄存器的 SHOCK[1:0]位用来设置 Shock 时间窗口值：这几个位的默认值为 00b，对应于 4/ODR 的时间，这里 ODR 为加速度计输出数据率。如果 SHOCK[1:0]位被置为其他不同的值，那么 1 LSB 对应于 8/ODR 的时间。

双击情况下，Quiet 时间窗口定义了第一次点击识别后的时间，期间不能发生超阈值。当锁存模式禁用（CTRL3 的 LIR 位置为 0）时，Quiet 时间还定义了中断脉冲的长度（单击和双击情况下均如此）。INT_DUR 寄存器的

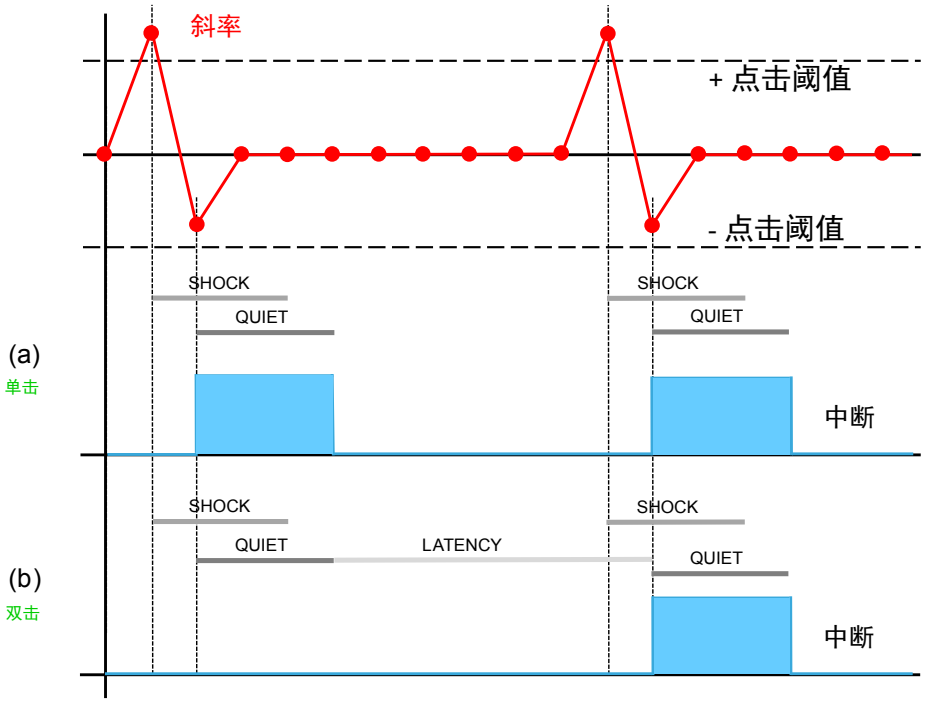
QUIET[1:0]位用来设置 Quiet 时间窗口值：这几个位的默认值为 00b，对应于 2/ODR 的时间，这里 ODR 为加速度计输出数据率。如果 QUIET[1:0]位被置为其他不同的值，那么 1 LSB 对应于 4/ODR 的时间。

双击情况下，latency 时间窗口定义了连续两次检测到点击之间的最大时间。latency 时间周期在第一次点击的 quiet 时间结束后开始。INT_DUR 寄存器的 LAT[3:0]位用来设置 latency 时间窗口值：这几个位的默认值为 0000b，对应于 16*ODR_XL 的时间，这里 ODR_XL 为加速度计输出数据率。如果 LAT[3:0]位被置为其他不同的值，那么 1 LSB 对应于 32/ODR 的时间。

图 10. 单击和双击识别（LIR 位 = 0） 显示了单击事件（a）和双击事件（b）。这些中断信号可被驱动至 INT1 中断引脚，单击情况下通过将 CTRL4 寄存器的 INT1_S_TAP 位置为 1 来实现，双击情况下通过将 CTRL4 寄存器的 INT1_TAP 位置为 1 来实现。

如果加速度计处于不活动状态，则不产生单击/双击中断（更多详细信息见 Section 5.7 活动/不活动识别）。

图 10. 单击和双击识别（LIR 位 = 0）



还可通过读取 TAP_SRC（38h）寄存器来检查点击中断信号，如表 13. TAP_SRC 寄存器所述。

表 13. TAP_SRC 寄存器

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

- 当检测到单击或双击事件时，TAP_IA 置为高电平。
- 当检测到单击时，SINGLE_TAP 置为高电平。
- 当检测到双击时，DOUBLE_TAP 置为高电平。
- TAP_SIGN 指示检测到点击事件时的加速度符号。符号为正时它为低电平，符号为负时它为高电平。
- 当在 X（Y、Z）轴上检测到点击事件时，X_TAP（Y_TAP、Z_TAP）置为高电平

单击和双击识别独立工作。将 WAKE_UP_THS 的 SINGLE_DOUBLE_TAP 位置为 0，则仅使能单击识别：双击识别被禁用，不能被检测到。当 SINGLE_DOUBLE_TAP 置为 1 时，单击和双击识别均使能。

如果锁存模式使能，且中断信号被驱动至中断引脚，则指定到 SINGLE_DOUBLE_TAP 的值还会影响中断信号的特性：当它被置为 0 时，单击中断信号可应用锁存模式；当它被置为 1 时，只有双击中断信号可应用锁存模式。锁存的中断信号保持为高电平，直至 TAP_SRC 寄存器被读取。如果使能了锁存模式，但是中断信号未驱动至中断引脚，那么锁存功能不起作用。

5.6.4 单击示例

下面的示例代码实现了用于单击检测的 SW 程序。

1.	将 60h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz, FS = ±2 *g*
2.	将 38h 写入 CTRL3	// 使能 X、Y、Z 轴上的点击检测
3.	将 09h 写入 TAP_6D_THS	// 设置点击阈值
4.	将 06h 写入 INT_DUR	// 设置 Quiet 和 Shock 时间窗口
5.	将 00h 写入 WAKE_UP_THS	// 只使能单击（SINGLE_DOUBLE_TAP = 0）
6.	将 40h 写入 CTRL4	// 单击中断驱动至 INT1 引脚

本例中，TAP_6D_THS 寄存器的 TAP_THS 字段被置为 01001b，因此点击阈值为 562.5 mg（= 9 * FS / 32）。

INT_DUR 寄存器的 SHOCK 字段被置为 10b：当斜率数据超出所编程阈值时，产生中断，并在 40 ms（= 2 * 8 / ODR）内返回到低于该阈值，这段时间对应于 Shock 时间窗口。

INT_DUR 寄存器的 QUIET 字段被置为 01b：由于锁存模式禁用，中断会保持高电平并持续 Quiet 窗口的时间，因此为 10 ms（= 1 * 4 / ODR）。

5.6.5 双击示例

下面给出的示例代码实现了用于单击检测的 SW 程序。

1.	将 60h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz, FS = ±2 *g*
2.	将 38h 写入 TAP_CFG	// 使能 X、Y、Z 轴上的点击检测
3.	将 0Ch 写入 TAP_6D_THS	// 设置点击阈值
4.	将 7Fh 写入 INT_DUR	// 设置 Duration、Quiet 和 Shock 时间窗口
5.	将 80h 写入 WAKE_UP_THS	// 使能单击 & 双击（SINGLE_DOUBLE_TAP = 1）
6.	将 08h 写入 CTRL4	// 双击中断驱动至 INT1 引脚

本例中，TAP_6D_THS 寄存器的 TAP_THS 字段被置为 01100b，因此点击阈值为 750 mg（= 12 * FS / 32）。

要实现中断生成，在第一次和第二次点击过程中，Shock 结束前，斜率数据必须返回到低于阈值。INT_DUR 寄存器的 SHOCK 字段被置为 11b，因此 Shock 时间为 60 ms（= 3 * 8 / ODR）。

对于中断生成，第一次点击识别后，在 Quiet 时间窗口内斜率数据不能超阈值。而且，由于锁存模式禁用，因此中断会保持高电平，并持续 Quiet 窗口的时间。INT_DUR 寄存器的 QUIET 字段被置为 11b，因此 Quiet 时间为 30 ms（= 3 * 4 / ODR）。

要使连续两次检测到的点击之间时间达到最大，INT_DUR 寄存器的 LAT 字段被置为 0111b，因此 Duration 时间为 560 ms（= 7 * 32 / ODR）。

5.7 活动/不活动识别

活动/不活动识别功能能够减少系统功耗，可支持开发新型智能应用。

当活动/不活动识别功能激活时，LIS2DS12 器件能够自动进入低功耗模式并将加速度计采样率降低至 12.5 Hz，当检测到唤醒中断事件时重新增加加速度计 ODR 和带宽。

利用此功能，根据用户所选的加速事件，系统可以高效地从低功耗模式转换成高性能模式，反之亦然，因此可以保证节能和灵活性。

通过将 WAKE_UP_THS 寄存器的 SLEEP_ON 位置为 1，可使能活动/不活动识别。

活动/不活动识别功能利用两个连续加速度采样之间的斜率来检测活动/不活动事件；斜率数据利用以下公式进行计算：

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

此功能可完全由用户编程，利用专门的寄存器组对所期望的斜率数据幅度和时序进行编程（图 11. 活动/不活动识别）。

该无符号阈值由 WAKE_UP_THS 寄存器的 WK_THS[5:0]位来定义；这些 6 位的 1 LSB 值取决于所选加速度计满量程：1 LSB = FS 的 1 / 64。该阈值可适用于正斜率数据和负斜率数据。

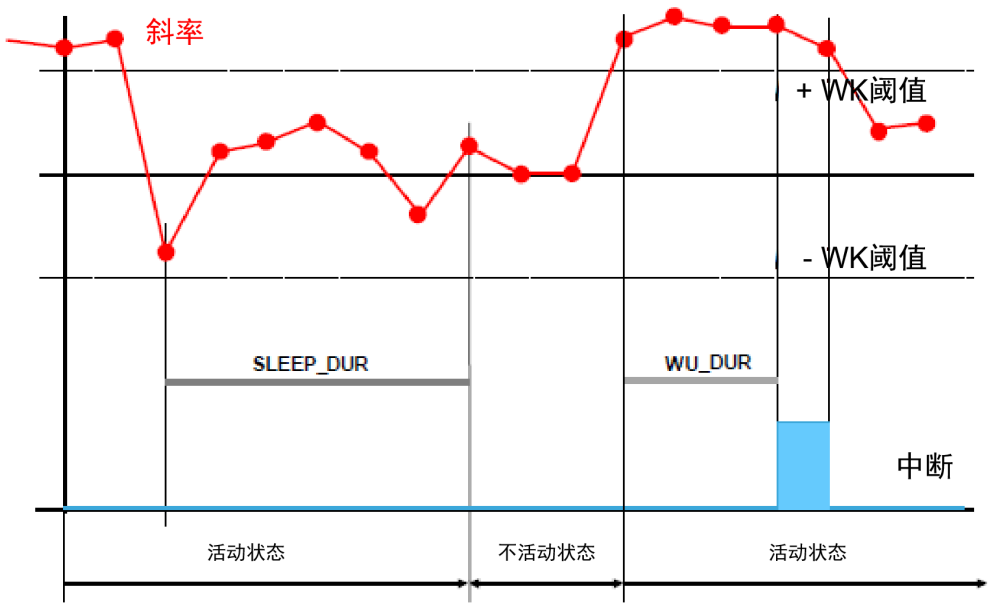
当一定数量的连续 X、Y、Z 斜率数据小于所配置阈值时，忽略 CTRL1 寄存器的 ODR [3:0]位（不活动），加速度计被内部地设置为 12.5 Hz，尽管 CTRL1 内容保持不变。待识别的不活动状态的持续时间由 WAKE_UP_DUR 寄存器的 SLEEP_DUR[3:0]位来定义：1 LSB 对应于 512/ODR 的时间，这里 ODR 为加速度计输出数据率。

当检测到不活动状态时，不会对应用处理器产生中断（WAKE_UP_SRC 寄存器的 SLEEP_STATE_IA 位不会传送到引脚）。

当一个轴上有一个采样的斜率数据大于阈值时，立即恢复 CTRL1 寄存器设置（活动）。唤醒中断事件可以通过 WAKE_UP_DUR 寄存器 WU_DUR [1:0]位的值来延迟：1 LSB 对应于 1/ODR 的时间，这里 ODR 为加速度计输出数据率。为了在不活动/活动事件的同时产生中断，WU_DUR [1:0]必须置为 0。

当检测到唤醒事件时，中断设置为高电平并持续 1/ODR 时长，然后自动置为无效（必须将 WU_IA 事件发送到引脚上，可通过将 CTRL4 寄存器的 INT1_WU 位设置为 1 来实现）。

图 11. 活动/不活动识别



下面给出的代码是实现活动/不活动检测的基本程序。

1.	将 50h 写入 CTRL1	// 启动加速度计 // ODR = 200 Hz, FS = ±2 *g*
2.	将 42h 写入 WAKE_UP_DUR	// 设置不活动检测的持续时间 // 设置唤醒检测的持续时间
3.	将 42h 写入 WAKE_UP_THS	// 设置活动/不活动阈值 // 使能活动/不活动检测
4.	将 20h 写入 CTRL4	// 活动（唤醒）中断驱动至 INT1 引脚

本例中，WAKE_UP_THS 寄存器的 WU_THS 字段被置为 000010b，因此活动/不活动阈值为 62.5 mg（= 2 * FS / 64）。

进行不活动检测前，X、Y、Z 斜率数据必须小于所配置阈值并持续一段时间，该时间由 WAKE_UP_DUR 寄存器的 SLEEP_DUR 字段定义：该字段被置为 0010b，对应 5.12 s（= 2 * 512 / ODR）。这段时间之后，加速度计 ODR 被内部地设置为 12.5 Hz。

如果（至少）一个轴的斜率数据大于阈值，并且在一段时间间隔之后通知发生唤醒中断，则会检测到活动状并立即恢复 CTRL1 寄存器设置，该时间间隔由 WAKE_UP_DUR 寄存器 WU_DUR 字段定义：此字段被置为 10b，对应 10 ms（= 2 * 1 / ODR）。

5.8 启动状态

器件上电后，LIS2DS12 执行一段 20 ms 的启动程序来加载修整参数。启动完成后，加速度计会自动配置为掉电模式。

上电后，可通过将 CTRL2 寄存器的 BOOT 位置为 1，来重载修调参数。

不需要切换设备电源线，器件控制寄存器内容不被修改，因此启动后器件工作模式不变。如果需要复位至控制寄存器的默认值，可通过将 CTRL2 寄存器的 SW_RESET 位置为 1 来实现。

通过将 CTRL5 寄存器的 INT2_BOOT 位置为 1，可以将启动状态信号驱动到 INT2 中断引脚：在进行启动时该信号变为'1'，完成后返回'0'。

要将器件恢复为掉电默认设置，在任何操作模式下都请执行以下步骤：

1. 将 SW_RESET 位置为"1"
2. 等待，直至 SW_RESET 位返回"0"
3. 将 BOOT 位置为"1"
4. 等待 20 ms

5.9 嵌入功能

LIS2DS12 器件在硬件中实现主要操作系统专用的传感器相关功能；功耗可忽略且高性能的专用 IP 模块可实现以下功能：

- 计步功能（步伐侦测和步数计算）
- 大幅运动检测
- 倾斜度检测

5.9.1 计步功能：步伐侦测和步数计算

LIS2DS12 器件的专用 IP 模块来专门实现计步功能：步伐侦测和步数计算。

计步功能工作于 25 Hz，因此加速度计 ODR 必须设置为 25 Hz 或更高的值。可以检测的最大计步频率约为 3 Hz。

为了启用计步功能，FUNC_CTRL 寄存器的 STEP_CNT_ON 位必须置为 1。当使能计步器时，MODULE_8BIT (0Ch)寄存器也可用，并包含加速强度计算($\sqrt{(x^2 + y^2 + z^2)}$)，最大更新频率为 25 Hz。

步伐侦测功能会在每次识别出一步时生成一个中断。如果在一定时间段内检测到了至少一步，即可生成中断，而不是每次识别出一步才生成中断。该时间段通过将高级配置寄存器中的 STEP_COUNT_DELTA 寄存器的位[0:7]置为一个高于 00h 的值来定义（STEP_COUNT_DELTA 寄存器的值的 1 LSB 对应于 1.6384 秒）。

在随机行走事件中，须检测到连续 7 步才能生成第一个中断（去抖动功能），以避免出现错误步数检测。

可以通过高级配置寄存器中的寄存器 PEDO_DEB_REG 的 DEB_STEP [2:0]位来修改去抖动步数。1LSB 对应 1 步。去抖动功能在一段时间（去抖动时间）后重启，并可通过高级配置寄存器中的寄存器 PEDO_DEB_REG 的 DEB_TIME[4:0]位来进行修改。1LSB 对应于 80 ms，默认值 = 13（13 *80 ms = 1040 ms）。

通过将 CTRL5 寄存器的 INT2_STEP_DET 位置为 1，可将该中断信号驱动至 INT2 中断引脚上；还可通过读取 FUNC_CK_GATE 寄存器的 STEP_DETECT 位对其进行检查。

使能计步功能后，计步器显示算法检测到的步数。步数由 STEP_COUNTER_H 和 STEP_COUNTER_L 寄存器级联给出，表示为一个 16 位无符号数字。当加速度计被配置为掉电或计步器禁用时，步数不会复位至 0；可通过将 STEP_COUNTER_MINTHS 寄存器的 RST_NSTEP 位置为 1 来将其复位至 0。

通过将 FIFO_CTRL（25h）的 INT2_STEP_COUNT_OV 位（位＃4）置为 1，可以将计步器溢出条件（计数达到 2^{16}）驱动到 INT2 引脚。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），则计步功能所产生的中断信号为脉冲的：中断引脚上观测到的脉冲持续时间约为 60 µs。

如果锁存模式使能（CTRL3 的 LIR 位置为 1），且中断信号被驱动至中断引脚，则当进行一步时，对 FUNC_CK_GATE 寄存器的读取会将两个引脚上的请求清除，器件准备好识别下一步。如果锁存模式使能但是该事件未驱动至中断引脚，那么 FUNC_CK_GATE 寄存器的 STEP_DETECT 位是脉冲的，且持续时间固定为 1/25 Hz。

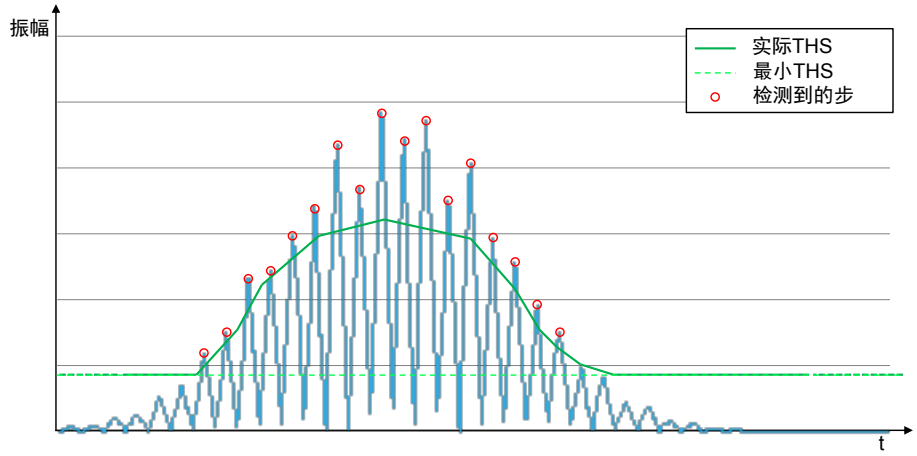
默认情况下，计步器工作的满量程为±2 *g*，与配置的器件满量程无关，但假如 CTRL1 寄存器中的满量程设置为至少±4 *g* 的话，可以通过将 STEP_COUNTER_MINTHS 寄存器的 PEDO4g 位置为 1 来将其配置为±4 *g*。

STEP_COUNTER_MINTHS 寄存器的 PEDO4g 位会影响 Module_8bit (0Ch)寄存器：如果将 PEDO4g 设为 1，则转换系数为 31.25 (mg/lsb)，否则为 15.625 (mg/lsb)。

还可以设置"最小阈值"，它是未检测到步数时步数识别阈值渐进趋向的值，并且不能下降到比它更小的值。此配置位于 STEP_COUNTER_MINTHS 寄存器的 SC_MTHS[5:0]字段。这 6 位的 1 LSB 的值取决于所选计步器满量程

(±2 *g* or ±4 *g*): 1 LSB = FS / 64.

图 12. 最小阈值



以下是一个基本 SW 程序，显示如何使能计步功能：

1. 将 20h 写入 CTRL1 // 启动加速度计 // ODR = 25 Hz, FS = ±2 *g*
2. 将 01h 写入 FUNC_CTRL // 使能计步算法
3. 将 04h 写入 CTRL5 // 步数探测器中断驱动至 INT2 引脚

当识别出一步时，产生该中断信号，可通过读取 STEP_COUNTER_H / STEP_COUNTER_L 寄存器来获取步数。

5.9.2 大幅运动检测

大幅运动检测功能可用于基于位置的应用，用来接收指示用户何时改变位置的通知。

当检测到可能是由于用户位置变化引起的“大幅运动”时，大幅运动功能会产生一个中断。要被认定为大幅运动，那么应至少运动了 5 步。

在 LIS2DS12 器件中，该功能已在硬件中使用加速度计实现，并以 25 Hz 工作，因此加速度计 ODR 必须设置为 25 Hz 或更高的值。

要启用大幅运动检测功能，FUNC_CTRL 寄存器的 SIGN_MOT_ON 位必须置为 1。当使能大幅运动检测时，Module_8bit (0Ch)寄存器也可用，并包含加速强度计算($\sqrt{(x^2 + y^2 + z^2)}$)，最大更新频率为 25 Hz。

默认情况下，大幅运动检测工作的满量程为±2 *g*，与配置的器件满量程无关，但假如 CTRL1 寄存器中的满量程设置为至少±4 *g* 的话，可以通过将 STEP_COUNTER_MINTHS 寄存器的 PEDO4g 位置为 1 来将其配置为±4 *g*。

STEP_COUNTER_MINTHS 寄存器的 PEDO4g 位会影响 Module_8bit (0Ch)寄存器：如果将 PEDO4g 设为 1，则转换系数为 31.25 (mg/lsb)，否则为 15.625 (mg/lsb)。

通过将 CTRL5 寄存器的 INT2_SIG_MOT 位置为 1，可将大幅运动中断信号驱动至 INT2 中断引脚上；还可通过读取 FUNC_CK_GATE 寄存器的 SIG_MOT_DETECT 位对其进行检查。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），大幅运动检测功能所产生的中断信号为脉冲的：中断引脚上观测到的脉冲持续时间约为 60 µs；FUNC_CK_GATE 寄存器的 SIG_MOT_DETECT 位上观测到的脉冲持续时间为 1/25 Hz。

如果锁存模式使能（CTRL3 的 LIR 位置为 1），且中断信号被驱动至中断引脚，则当检测到“大幅运动”时，对 FUNC_CK_GATE 寄存器的读取会将 INT2 上的请求以及 FUNC_CK_GATE 寄存器的 SIG_MOT_DETECT 位清除，器件准备好识别下一事件。如果锁存模式使能但是中断信号未驱动至中断引脚，那么 FUNC_CK_GATE 寄存器的 SIG_MOT_DETECT 位是脉冲的，且持续时间固定为 1/25 Hz。

用户可配置大幅运动最小检测阈值，该阈值是产生大幅运动中断前，位置改变时用户所行走的步数。为此，可以使用高级配置寄存器的 SM_THS 寄存器（34h）：

位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
SM_THS_7	SM_THS_6	SM_THS_5	SM_THS_4	SM_THS_3	SM_THS_2	SM_THS_1	SM_THS_0

请注意，当 SMD 阈值（SM_THS（34h）寄存器的 SM_THS_ [7:0]位）等于或高于计步器去抖动阈值（PEDO_DEB_REG（2Bh）高级寄存器的 DEB_STEP [2:0]位）时，该 SMD 阈值有效。下表显示了 SMD 阈值的可能情况。

表 14. SMD 阈值

配置	SM_THS_[7:0]	DEB_STEP[2:0]
计步器去抖动未激活	√	
计步器去抖动激活，SMD 阈值 ≥ 计步器去抖动阈值	√	
计步器去抖动激活，SMD 阈值 < 计步器去抖动阈值		√

当计步器去抖动激活且 SMD 阈值低于默认值（= 6）时，必须相应地减小 PEDO_DEB_REG 寄存器的位 DEB_STEP [2:0]。

请注意：过度减少计步器去抖动阈值可能导致计步器步数检测报错！

下面是显示如何使能大幅运动检测功能的基本 SW 程序：

1. 将 20h 写入 CTRL1

```
// 启动加速度计
// ODR = 25 Hz, FS = ±2 g
```

2.	将 02h 写入 FUNC_CTRL	// 启用大幅运动检测算法
3.	设置 CTRL2 中的 10h 位	// 启用对高级配置寄存器的访问： // 将寄存器 21h 的位#4 写入 1，其他位保持不变

4.	在 SM_THS[7:0]中设置阈值	// 在高级配置寄存器的 34h 中写入阈值。
		// 1LSB = 1 步。默认值 = 6。
		// 禁用对高级配置寄存器的访问。

5.	将 00h 写入 CTRL2	// 将 reg 3Fh 的位#4 写入 0。 // 请注意，所有位都是只读的，第 4 位除外。
6.	将 08h 写入 CTRL5	// 大幅运动中断驱动至 INT2 引脚

5.9.3 倾斜度检测

倾斜度检测功能支持检测何时发生活动改变（例如，当电话在前口袋中用户从坐到站或从站到坐时）：在 LIS2DS12 器件中，可在硬件中实现它。

要启用倾斜度检测，FUNC_CTRL 寄存器的 TILT_ON 位必须置为 1。

如果器件配置为用于倾斜度检测，那么当器件距起始位置的倾斜角度大于 35 度时，会产生一个中断。起始位置定义为倾斜检测使能时器件的位置，或上一次倾斜中断产生时的器件位置。

使能此功能后，要产生第一次倾斜中断，器件距起始位置的倾斜角度应大于 35 度并持续至少 2 秒的时间。产生第一次倾斜中断后，当器件距离上次中断检测时的器件位置的倾斜角度大于 35 度时（不需要等待 2 秒），倾斜中断信号即被置为高电平。

通过将 CTRL5 寄存器的 INT2_TILT 位置为 1，可将该中断信号驱动至 INT2 引脚上；还可通过读取 FUNC_CK_GATE 寄存器的 TILT_INT 位对其进行检查。

如果锁存模式禁用（CTRL3 的 LIR 位置为 0），倾斜度检测功能所产生的中断信号为脉冲的：中断引脚上观测到的脉冲持续时间约为 60 µs；FUNC_CK_GATE 寄存器的 TILT_INT 位上观测到的脉冲持续时间为 1/25 Hz。

如果锁存模式使能（CTRL3 的 LIR 位置为 1），且中断信号被驱动至中断引脚，那么当检测到倾斜时，对 FUNC_CK_GATE 寄存器的读取会将 INT2 引脚上的请求以及 FUNC_CK_GATE 寄存器的 TILT_INT 位清零，器件准备识别下一次倾斜事件。如果锁存模式使能但是中断信号未驱动至中断引脚，那么 FUNC_CK_GATE 寄存器的 TILT_INT 位是脉冲的，且持续时间固定为 1/25 Hz。

倾斜度检测功能工作于 25 Hz，独立于器件 ODR。

下面是显示如何使能倾斜度检测功能的基本 SW 程序：

1.	将 20h 写入 CTRL1	// 启动加速度计 // ODR = 25 Hz, FS = ±2 *g*
2.	将 10h 写入 FUNC_CTRL	// 使能倾斜度检测
3.	将 10h 写入 CTRL5	// 倾斜探测器中断驱动至 INT2 引脚

5.10 传感器集线器

传感器集线器功能允许将一个外部传感器连接到 LIS2DS12 器件的 I^2C 主接口。外部传感器数据不进入 FIFO，必须直接从传感器集线器寄存器进行读取。

传感器必须连接到器件的 SDx/SCx 引脚。上拉电阻可以通过 FUNC_CTRL（3Fh）中的 TUD_EN 来内部使能，或从外部添加。

如果加速度计处于掉电模式，则传感器集线器不工作。传感器集线器 I^2C 活动由加速度计 ODR 触发。当传感器集线器使能时，每次触发器被激活，都会在总线上持续生成配置的 I^2C 读/写事务。即使将加速度计 ODR 设置为较高值，触发器频率也不会超过 100 Hz。

5.10.1 传感器集线器引脚说明

当传感器集线器使能时，一些标准引脚自动复用，以执行不同的功能。

表 15. 传感器集线器引脚说明

引脚号	引脚名称	传感器集线器功能	说明
3	SDO SA0	MSDA	I²C 主数据线
11	INT2	MSCL	I²C 主时钟线

如表 15. 传感器集线器引脚说明所述，传感器集线器使用引脚 3 作为 I²C 主数据线，使用引脚 11 作为 I²C 主时钟线，并必须作以下限制：

1. 当 LIS2DS12 器件通过 SPI 总线连接到应用处理器时，必须使用 SPI 3 线（SDO 不可用）。CTRL2（21h）中的 SIM 位必须置为 1。
2. 当 LIS2DS12 器件通过 I²C 总线连接到应用处理器时，SA0 引脚不可用（不被传感器集线器使用），并且它内部默认为“0”。因此，当传感器集线器启用时，只有一个（而不是两个）LIS2DS12 器件可以连接到应用处理器（采用 I²C 7 位地址格式 1Eh）。

5.10.2 配置传感器集线器

如果操作是读取或写入以及要读取的字节数（读取操作）或要写入的字节（写入操作），则用户可以通过提供从设备地址、要访问的寄存器地址来配置传感器集线器。高级配置寄存器（当 CTRL2 寄存器的 FUNC_CFG_EN 位被置为 1 时可访问）用来配置关联到外部传感器的 I²C 从线接口，下面对此进行介绍。

表 16. SLV0_ADD (30h)寄存器

b7	b6	b5	b4	b3	b2	b1	b0
Slave0_ add6	Slave0_ add5	Slave0_ add4	Slave0_ add3	Slave0_ add2	Slave0_ add1	Slave0 add0	rw_0

- Slave0_add[6:0]位用来指示外部传感器的 I²C 从线地址。
- rw_0 位对外部传感器进行读取/写入操作（0：写操作；1：读操作）。当发生下一个传感器集合（sensor hub）触发事件时，执行读取/写入操作。

表 17. SLV0_SUBADD (31h)寄存器

b7	b6	b5	b4	b3	b2	b1	b0
Slave0_ reg7	Slave0_ reg6	Slave0_ reg5	Slave0_ reg4	Slave0_ reg3	Slave0_ reg2	Slave0_ reg1	Slave0_ reg0

- Slave0_reg[7:0]位用来指示要写入的外部传感器寄存器的地址（如果 SLV0_ADD 寄存器的 rw_0 位置为 0）或要读取的外部寄存器的地址（如果 SLV0_ADD 寄存器的 rw_0 位被置为 1）。

表 18. SLV0_CONFIG (32h)寄存器

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	Slave0_ numop2	Slave0_ numop1	Slave0_ numop0

- Slave0_numop[2:0]位专门用来定义从 SLV0_SUBADD 寄存器所示的寄存器地址开始的第一个外部传感器上执行的连续读取操作数。

表 19. DATAWRITE_SLV0 (33h)寄存器

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

- 当 SLV0_ADD 寄存器的 rw_0 位被置为 0（写操作）时，Slave_dataw[7:0]位专门用来表示要写入到外部传感器（其地址在 SLV0_SUBADD 寄存器中指定）的数据。

LIS2DS12 器件在 I^2C 时钟和数据线上提供可选的内部上拉电阻，以实现适当的 I^2C 功能。要使能内部上拉功能，用户必须设置 FUNC_CTRL 寄存器的 TUD_EN 位。

表 20. FUNC_CTR（3Fh）寄存器

b7	b6	b5	b4	b3	b2	b1	b0
-	-	MODULE _ON	TILT _ON	TUD _ON	主 _ON	SIG_MOT _ON	STEP_ CNT_ON

5.10.3 使能传感器集线器

通过将寄存器 FUNC_CTRL 中的 MASTER_ON 位置为逻辑“1”来开启传感器集线器功能。

表 21. FUNC_CTR（3Fh）寄存器

b7	b6	b5	b4	b3	b2	b1	b0
-	-	MODULE_ON	TILT_ON	TUD_ON	主_ON	SIG_MOT_ON	STEP_CNT_ON

当传感器集线器使能，就会根据高级配置寄存器的用户配置开始在总线上生成 I²C 事务。它会在给定配置下进行循环，直至传感器集线器禁用（MASTER_ON 为逻辑“0”）。在每次循环开始时，传感器集线器将 FUNC_SRC 寄存器的 SENSORHUB_END_OP 位置为逻辑“0”。然后它启动 I²C 事务，最后将 SENSORHUB_END_OP 位置为逻辑“1”。

表 22. FUNC_SRC（3Eh）寄存器

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	-	-	RST_TILT	MODULE_READY	SENSORHUB_END_OP

此外，通过将 CTRL4 寄存器的 INT1_MASTER_DRDY 位置为“1”，指示何时完成传感器集线器操作的位能够可选地发送到 INT1 上。

表 23. CTRL4 (23h)寄存器

b7	b6	b5	b4	b3	b2	b1	b0
INT1_MASTER_DRDY	INT1_S_TAP	INT1_WU	INT1_FF	INT1_TAP	INT1_6D	INT1_FTH	INT1_DRDY

5.10.4 从传感器集线器读取采样

当辅助 I²C 主线使能时，外部传感器可读取若干寄存器，该数量等于 Slave0_numop 字段的值，从 SLV0_SUBADD 寄存器指定的寄存器地址开始。读取的数据会连续存储在从 SENSORHUB1_REG 开始的寄存器中（存储顺序与读取顺序相同）。所以最多可以读取 6 个字节。

表 24. 传感器集线器寄存器

寄存器名	地址	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
SENSORHUB1_REG	06h	SHub1_7	SHub1_6	SHub1_5	SHub1_4	SHub1_3	SHub1_2	SHub1_1	SHub1_0
SENSORHUB2_REG	07h	SHub2_7	SHub2_6	SHub2_5	SHub2_4	SHub2_3	SHub2_2	SHub2_1	SHub2_0
SENSORHUB3_REG	08h	SHub3_7	SHub3_6	SHub3_5	SHub3_4	SHub3_3	SHub3_2	SHub3_1	SHub3_0
SENSORHUB4_REG	09h	SHub4_7	SHub4_6	SHub4_5	SHub4_4	SHub4_3	SHub4_2	SHub4_1	SHub4_0
SENSORHUB5_REG	0Ah	SHub5_7	SHub5_6	SHub5_5	SHub5_4	SHub5_3	SHub5_2	SHub5_1	SHub5_0
SENSORHUB6_REG	0Bh	SHub6_7	SHub6_6	SHub6_5	SHub6_4	SHub6_3	SHub6_2	SHub6_1	SHub6_0

5.10.5 传感器集线器示例

本节提供了关于如何使用传感器集线器的示例，假设使用 LIS3MDL 磁力计作为外部传感器。本示例还假定 LIS2DS12 内部加速度计可在任意 ODR 值下启用。

- 读取 LIS3MDL WhoAmI 寄存器

/* 配置高级配置寄存器 *

1. 将 15h 写入 21h // 将 FUNC_CFG_EN 置为 1，以便访问 // 高级配置寄存器
2. 将 3Dh 写入 30h // SLV0_ADD + R 模式下，LIS3MDL I2C 地址
3. 将 0Fh 写入 31h // SLV0_SUBADD 中 WHO_AM_I 地址
4. 将 01h 写入 32h // 读取 1 字节
5. 将 00h 写入 3Fh // 将 FUNC_CFG_EN 置为 0，以禁用对高级配置寄存器的访问。请注意，所有位都是只读的，第 4 位除外。

/* 控制传感器集线器活动 */

6. 将 0Ch 写入 3Fh // 使能传感器集线器 + 上拉
7. 等待 3Eh 中 bit0 等于 1 // SENSORHUB_END_OP 等于 1
8. 读取 06h // WAI 寄存器在 SENSORHUB1_REG 中
9. 将 00h 写入 3Fh // 禁用传感器集线器

- 编程 LIS3MDL 器件

/* 配置高级配置寄存器 *

1. 将 15h 写入 21h // 将 FUNC_CFG_EN 置为 1，以便访问 // 高级配置寄存器
2. 将 3Ch 写入 30h // SLV0_ADD + W 模式下，LIS3MDL I2C 地址
3. 将 22h 写入 31h // SLV0_SUBADD 中的 CTRL3 寄存器地址
4. 将 01h 写入 32h // 写入 1 字节
5. 将 00h 写入 33h // LIS3MDL 处于连续转换模式
6. 将 00h 写入 3Fh // 将 FUNC_CFG_EN 置为 0，以禁用对高级配置寄存器的访问。请注意，所有位都是只读的，第 4 位除外。

/* 控制传感器集线器活动 */

7. 将 0Ch 写入 3Fh // 使能传感器集线器 + 上拉
8. 等待 3Eh 中 bit0 等于 1 // SENSORHUB_END_OP 等于 1
9. 将 00h 写入 3Fh // 禁用传感器集线器

- 读取 LIS3MDL 采样

/* 配置高级配置寄存器 *

1. 将 15h 写入 21h // 将 FUNC_CFG_EN 置为 1，以便访问 // 高级配置寄存器
2. 将 3Dh 写入 30h // SLV0_ADD + R 模式下，LIS3MDL I2C 地址
3. 将 28h 写入 31h // LIS3MDL OUT_X_L 寄存器地址
4. 将 06h 写入 32h // 读取 6 字节

5. 将 00h 写入 3Fh	// 将 FUNC_CFG_EN 置为 0，以禁用对高级配置寄存器的访问。请注意，所有位都是只读的，第 4 位除外。
/* 控制传感器集线器活动 */	
6. 将 0Ch 写入 3Fh	// 使能传感器集线器 + 上拉
7. 等待 3Eh 中 bit0 等于 1	// SENSORHUB_END_OP 等于 1
8. 读取 06h	// SENSORHUB1_REG 中的 X_L
9. 读取 07h	// SENSORHUB1_REG 中的 X_H
10. 读取 08h	// SENSORHUB1_REG 中的 Y_L
11. 读取 09h	// SENSORHUB1_REG 中的 Y_H
12. 读取 0Ah	// SENSORHUB1_REG 中的 Z_L
13. 读取 0Bh	// SENSORHUB1_REG 中的 Z_H

只要传感器集线器使能，它就会一直从 LIS3MDL 读取数据，速度为触发频率（最大为 100 Hz）。

6 先进先出（FIFO）缓冲区

为了限制主机处理器干预并便于对事件识别数据进行后处理，LIS2DS12 为三条输出通道 X、Y 和 Z 分别嵌入了先进先出（FIFO）缓冲区。

使用 FIFO 可使系统实现一致的节能效率，仅当需要时才会唤醒 FIFO，并会从 FIFO 批量输出重要数据。

FIFO 缓冲区可在五种不同模式下工作，各个模式可确保在应用开发过程中实现高度灵活性：Bypass 模式、FIFO 模式、Continuous 模式、Bypass-Continuous 模式和 Continuous-FIFO 模式。

可使能可编程水印等级和 FIFO_FULL 事件在 INT1 引脚上生成专用中断。

6.1 FIFO 描述

FIFO 缓冲区能够为每个通道存储多达 256 个 14 位的加速度采样，或是存储多达 768 个加速度模计算输出（参见 Section 6.4 模至 FIFO）；数据存储为 14 位的二进制补码左对齐表示，这意味着它们必须被右移两位。

数据样本集合由 6 个字节（XI、Xh、YI、Yh、ZI 和 Zh）和组成，它们会以选定的输出数据速率（ODR）释放到 FIFO 中。

新样本集合会放在第一个空闲的 FIFO 位置中，缓冲区被占满后，新样本集合会覆盖最早的值。

表 25. FIFO 缓冲区完整表示（存储了第 256 个样本集合）

输出寄存器	28h	29h	2Ah	2Bh	2Ch	2Dh
	XI	Xh	YI	Yh	ZI	Zh
FIFO 索引	FIFO 样本集合					
FIFO（0）	XI（0）	Xh（0）	YI（0）	Yh（0）	ZI（0）	Zh（0）
FIFO（1）	XI（1）	Xh（1）	YI（1）	Yh（1）	ZI（1）	Zh（1）
FIFO（2）	XI（2）	Xh（2）	YI（2）	Yh（2）	ZI（2）	Zh（2）
FIFO（3）	XI（3）	Xh（3）	YI（3）	Yh（3）	ZI（3）	Zh（3）
...	...	...	...	...	...	...
FIFO（254）	XI（254）	Xh（254）	YI（254）	Yh（254）	ZI（254）	Zh（254）
FIFO（255）	XI（255）	Xh（255）	YI（255）	Yh（255）	ZI（255）	Zh（255）

表 26. FIFO 缓冲区完整表示（存储了第 257 个样本集合、丢弃了第 1 个样本）

输出寄存器	28h	29h	2Ah	2Bh	2Ch	2Dh
	XI	Xh	YI	Yh	ZI	Zh
FIFO 索引	样本集合					
FIFO（0）	XI（1）	Xh（1）	YI（1）	Yh（1）	ZI（1）	Zh（1）
FIFO（1）	XI（2）	Xh（2）	YI（2）	Yh（2）	ZI（2）	Zh（2）
FIFO（2）	XI（3）	Xh（3）	YI（3）	Yh（3）	ZI（3）	Zh（3）
FIFO（3）	XI（4）	Xh（4）	YI（4）	Yh（4）	ZI（4）	Zh（4）
...	...	...	...	...	...	...
FIFO（255）	XI（256）	Xh（256）	YI（256）	Yh（256）	ZI（256）	Zh（256）

表 25. FIFO 缓冲区完整表示（存储了第 256 个样本集合） 表示的是存储了 256 个样本时 FIFO 已满的状态，而表 26. FIFO 缓冲区完整表示（存储了第 257 个样本集合、丢弃了第 1 个样本）表示的是第 257 个样本插入到 FIFO 中、第 1 个样本被覆盖时的下一步。新的最早样本集合在输出寄存器中可用。

如果 FIFO 已使能，并且所处模式不是 Bypass 模式，LIS2DS12 输出寄存器（28h 到 2Dh）始终会包含最早的 FIFO 样本集合。

6.2 FIFO 寄存器

FIFO 缓冲区由四个不同的加速度计寄存器进行管理，其中两个寄存器可使能并配置 FIFO 特性，另外两个寄存器会提供关于缓冲区状态的信息。

一些其他寄存器用于传送引脚上的 FIFO 事件以中断应用处理器。这些在 Section 6.3 FIFO 中断中进行讨论。

6.2.1 FIFO_CTRL (25h)

FIFO_CTRL 寄存器包含有设置 FIFO 的模式。默认复位时，FIFO 模式为 Bypass，表示关闭；只要模式设置为 Bypass 以外的其他模式，FIFO 就会启用并开始存储采样值（或模值）。

表 27. **FIFO_CTRL 寄存器**

b7	b6	b5	b4	b3	b2	b1	b0
FMODE2	FMODE1	FMODE0	INT2_ STEP_ COUNT _OV	MODULE_TO _FIFO	0	0	IF_CS_ PU_DIS

FMODE[2:0]位可选择 FIFO 缓冲区行为：

1. FMODE[2:0] = 000b：Bypass 模式（FIFO 关闭）
2. FMODE[2:0] = 001b：FIFO 模式
3. FMODE[2:0] = 011b：Continuous-FIFO 模式
4. FMODE[2:0] = 100b：Bypass-Continuous 模式
5. FMODE[2:0] = 110b：Continuous 模式

MODULE_TO_FIFO 使 X/Y/Z 采样（sqrt（$x^2+y^2+z^2$））的模值存入 FIFO 中，而非存入采样值本身。请注意，FUNC_CTRL 寄存器的 MODULE_ON 位必须启用。当 MODULE_ON 位置为 1 时，嵌入功能（计步、倾斜和大幅运动检测）不可用。

6.2.2 FIFO_THS (2Eh)

该寄存器可以用来设置 FIFO 阈值等级。

表 28. **FIFO_THS 寄存器**

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

FTH [7:0]位定义水印等级；当 FIFO 内容大于或等于此值时，FIFO_SRC 寄存器中的 FTH 位置为“1”。

6.2.3 FIFO_SRC (2Fh)

该寄存器每个 ODR 会更新一次，会提供关于 FIFO 缓冲区状态的信息。

表 29. FIFO_SRC 寄存器

b7	b6	b5	b4	b3	b2	b1	b0
FTH	FIFO_ OVR	DIFF8	0	0	0	0	0

- 如果 FIFO 内容超过水印等级，FTH 位会置为高电平。此标志可发送至引脚（参见 Section 6.3 FIFO 中断）。
- 在 FIFO 缓冲区满后，重写第一个采样时，FIFO_OVR 位置为高电平。这意味着 FIFO 缓冲区包含 256 个未读采样。第一个样本集合已被读取时，FIFO_OVR 位会复位。
- DIFF8 位（或 FIFO_FULL 位）与位 FIFO_SAMPLES（DIFF[7:0]）一起使用，可提供用了多少 FIFO 空间的信息（000000000b 表示 FIFO 为空，100000000b 表示 FIFO 已满）。此标志可发送至引脚（参见 Section 6.3 FIFO 中断）。

寄存器内容会与 FIFO 写操作和读操作同步更新。

表 30. FIFO_SRC 特性（假定 FTH[7:0] = 15）

FTH	DIFF8 （FIFO_FULL）	FIFO_OVR	DIFF[8:0]	未读 FIFO 样本	时序
0	0	0	000000000	0	t0
0	0	0	000000001	1	t0 + 1/ODR
0	0	0	000000010	2	t0 + 2/ODR
...	...	...	...	...	...
0	0	0	000001110	14	t0 + 14/ODR
1	0	0	000001111	15	t0 + 15/ODR
...	...	...	...	...	...
1	0	0	011111111	255	t0 + 255/ODR
1	1	0	100000000	256	t0 + 256/ODR
1	1	1	100000000	256	t0 + 257/ODR

6.2.4 FIFO_SAMPLES (30h)

该寄存器的内容与 FIFO_SRC 寄存器（DIFF [7:0]）的 DIFF8 位一起使用，可提供用了多少 FIFO 空间的信息（000000000b 表示 FIFO 为空，100000000b 表示 FIFO 已满）。

表 31. FIFO_SAMPLES 寄存器

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

6.3 FIFO 中断

有两个特定的 FIFO 事件可以传输到引脚，以中断主处理器：FIFO 阈值和 FIFO 已满。

第三个 FIFO 事件，FIFO_OVR，不会被传送到引脚，而是通过读取 FIFO_SRC 寄存器的相应位来进行轮询。

6.3.1 FIFO 阈值

FIFO 阈值是可用于生成特定中断的可配置功能，可用于确定 FIFO 缓冲区何时包含的样本数至少为定义为阈值等级的数目。用户可以使用 FIFO_THS 寄存器中的 FTH [7:0]字段在 0 到 255 之间的范围内选择所需的等级。

如果 FIFO（DIFF [8:0]）中的条目数大于或等于 FTH [7:0]中编程的值，则 FIFO_SRC 寄存器中的 FTH 位置为高电平。

DIFF[8:0]会以 ODR 频率增加一步，每次由用户执行样本集合读取操作时，DIFF[8:0]会减小一步。

阈值标志（FTH）可以传送到 INT1 和 INT2 引脚，为应用处理器提供专用中断，使每次中断之间功耗更少。CTRL4 寄存器的 INT1_FTH 位和 CTRL5 寄存器的 INT2_FTH 位专门用于此目的。

6.3.2 FIFO 已满

只要 FIFO 已满，就可以配置器件，使之产生一个中断。为此，只需将 WAKE_UP_DUR 寄存器的 INT1_FSS7 位置为“1”即可。为避免丢失样本，FIFO 读取操作必须在 1 个 ODR 窗口内开始并完成。

6.4 模至 FIFO

如果模计算开启（FUNC_CTRL 寄存器的 MODULE_ON 位为“1”），且 FIFO_CTRL 的 MODULE_TO_FIFO 位置为“1”，则 FIFO 缓冲区中不是加速度采样，而是其模计算值（$\sqrt{（x^2+y^2+z^2）}$）。

由于模大小为 14 位，所以 FIFO 缓冲区最多可能包含 768 个模值（256 * 3）。当模值存储在 FIFO 中时，FIFO_THS 中设置的阈值和 FIFO_SAMPLES 寄存器中可用的存储数据数量由 1LSb = 3 样本表示。

如果使能了其中一个嵌入功能，则无法使能模至 FIFO 功能（计步器、大幅运动、倾斜）。模至 FIFO 的最大输出传感器数据率（ODR）为 800 Hz。

以下是使 FIFO 中存入加速度采样模值的简单过程：

1. 在 FUNC_CTRL（3Fh）中将 MODULE_ON 置为 1，使能模计算
2. 将 FIFO_CTRL（25h）的 MODULE_TO_FIFO 位置为“1”，使模值保存在 FIFO 缓冲区中
3. 在其中一个模式下使能 FIFO（参见 Section 6.5 FIFO 模式）

6.5 FIFO 模式

通过 FIFO_CTRL 寄存器的 FMODE[2:0]字段，LIS2DS12 FIFO 缓冲器可配置为五种不同的可选工作模式。可用配置确保了高度灵活性，并扩展了可用于应用开发的功能数量。

以下段落描述了 Bypass、FIFO、Continuous、Bypass-Continuous 和 Continuous-FIFO 模式。

6.5.1 Bypass 模式

启用 Bypass 模式后，FIFO 不可运行：缓冲区内容会被清空、输出寄存器（0x28 到 0x2D）会冻结为最后载入的值，在选择其他模式之前，FIFO 缓冲区会保持空白状态。

可以通过在 FIFO_CTRL 寄存器中将 FMODE [2:0]字段置为 000b 来激活 Bypass 模式。

当在不同模式下工作时，要停止和复位 FIFO 缓冲器，必须使用 Bypass 模式。请注意，将 FIFO 缓冲区置于 Bypass 模式会清除整个缓冲区的内容。

6.5.2 FIFO 模式

FIFO 模式中，缓冲区继续填充直至充满（存储了 256 个样本集合）。一旦 FIFO_OVR 标志变为“1”，FIFO 就停止采集数据，其内容保持不变，直至选择不同的模式。

可以通过在 FIFO_CTRL 寄存器中将 FMODE [2:0]字段置为 001b 来激活 FIFO 模式。

选择该模式后，FIFO 会开始进行数据采集，DIFF[8:0]也会根据存储的样本数发生变化。程序结束时，FIFO_OVR 标志上升为 1，然后可以恢复数据，从输出寄存器读取 256 个样本集合。由于在 FIFO 模式下数据采集已停止，并且不存在覆盖已获取数据的风险，因此通信速度并不重要。重新启动 FIFO 模式之前，请务必在读取程序之后退出 Bypass 模式。

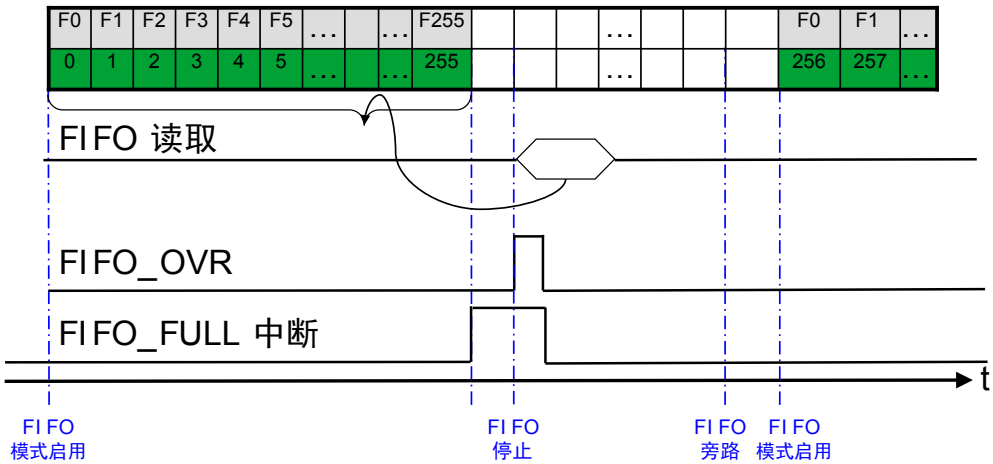
要尽快处理 FIFO_FULL 事件，建议将其传送到引脚，以产生一个中断，该中断随后会由一个特定的处理程序进行管理：

1. 将 INT1_FSS7 置为‘1’：使能 FIFO_FULL 中断
2. 设置 FMODE[2:0] = 001b：使能 FIFO 模式

当生成了 FIFO_FULL 中断或 FIFO_OVR 位为高电平时（轮询模式）：

1. 从加速度计输出寄存器读取数据

图 13. FIFO 模式特性



如图 13. FIFO 模式特性中所示，当使能了 FIFO 模式，缓冲区会开始采集数据，并会以所选输出数据速率填入全部 256 个位置（从 F0 到 F255）。缓冲区已满后，下一采样会进入并重写缓冲区，FIFO_OVR 位会变为高电平，数据采集会永久停止；用户可随时读取 FIFO 内容，因为在选择 Bypass 模式之前，FIFO 缓冲区的内容保持不变。读取程序可以在由 FIFO_FULL 条件（DIFF8）触发的中断处理程序内执行，它包括 256 个 6 字节样本集合（共 1236 字节），会从 FIFO 中存储的最早样本（F0）开始获取数据。第一个样本集合已被读取时，FIFO_OVR 位会复位。Bypass 模式设置会复位 FIFO 并允许用户再次使能 FIFO 模式。

6.5.3 连续模式

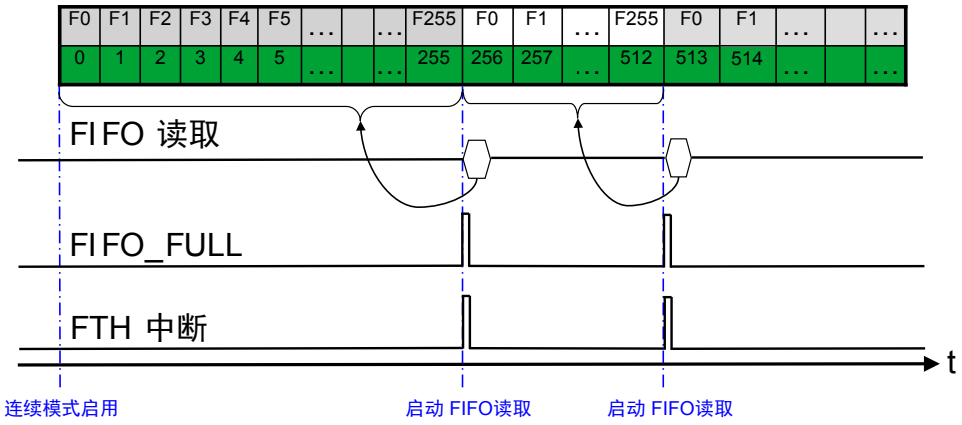
在 Continuous 模式下，FIFO 会继续填入数据，如果缓冲区已满，FIFO 索引会从头开始使用，较早的数据会被当前数据替代。最早先的数据继续被覆盖，直至读取操作释放了 FIFO 插槽。为了使 FIFO 位置的释放速度快于获得新数据的速度，主机处理器的读取速度至关重要。Bypass 配置下 FMODE[2:0]用于停止该模式。

按照以下步骤进行 FIFO Continuous 配置，其中设置一个阈值来产生中断，以触发应用处理器进行读取：

1. 将 FTH[7:0]置为 255。
2. 将 INT2_FTH 置为‘1’：使能 FIFO 阈值中断
3. 将 FIFO_CTRL 寄存器（25h）中的 FMODE[2:0]字段设为 110b 可激活 Continuous 模式。

当产生 FTH 中断时，从加速度计输出寄存器读取数据。

图 14. 带中断触发的 Continuous 模式



如图 14. 带中断触发的 Continuous 模式所示，当使能了 Continuous 模式时，FIFO 缓冲区会以选定的输出数据速率持续填入数据（从 F0 到 F255）。当缓冲区满时，FTH 中断（以及 FIFO_SRC（2Fh）中的 DIFF8 位指示的 FIFO_FULL 条件，该条件也可用于触发中断）变为高电平，应用处理器会立即读取所有 FIFO 样本（256 * 6 字节），以免丢失数据并限制主机处理器的干预，从而提高系统效率。关于 FIFO 读取速度的更多详细信息，请参见 Section 6.6 从 FIFO 读取数据。

读取命令发送到器件后，输出寄存器内容会移动到 SPI/I^2C 寄存器，当前最早的 FIFO 值会移入输出寄存器，以执行下一次读取操作。

6.5.4 连续-FIFO 模式

此模式是先前所述的连续和 FIFO 模式的组合。在 Continuous-FIFO 模式下，FIFO 缓冲区会在 Continuous 模式下开始工作，并会在发生选定的中断（例如，唤醒、自由落体、点击……）后切换为 FIFO 模式。

可使用此模式来分析生成中断的样本历史；标准操作是在 FIFO 模式已触发、FIFO 缓冲区已满并停止时读取 FIFO 内容。

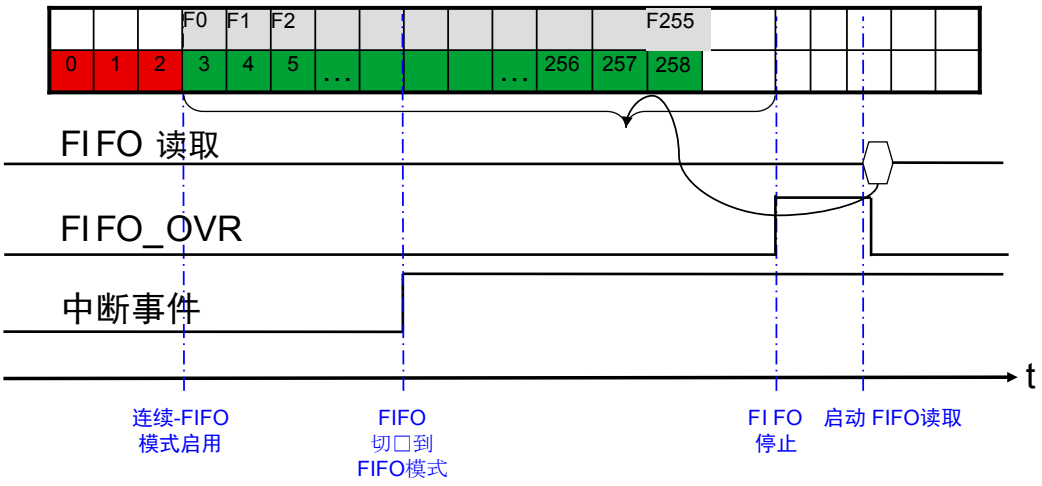
请按照以下步骤配置 Continuous-FIFO 模式：

1. 按照 Section 5 中断生成和嵌入功能中说明配置所需的中断发生器（确保锁定）。
2. 将 FIFO_CTRL 寄存器（25h）中的 FMODE[2:0]字段设为 011b 可激活 Continuous-FIFO 模式。

注意：当发生所请求的事件时，当且仅当事件标志被发送到 INT1 或 INT2 引脚时，才会触发 FIFO 模式更改。

在 Continuous 模式下，FIFO 缓冲区持续填充；当请求的事件发生时，FIFO 模式发生变化；然后，一旦缓冲器变满，FIFO_OVR 位置为高电平，下一个采样将会覆盖早先数据，FIFO 停止采集数据（见图 15. Continuous-FIFO 模式：中断锁存和非锁存）。

图 15. Continuous-FIFO 模式：中断锁存和非锁存



6.5.5 旁路-连续模式

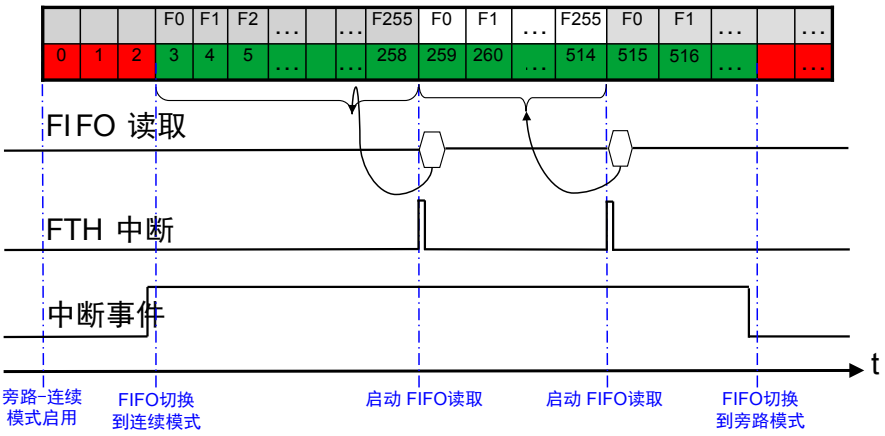
此模式是先前所述的旁路和连续模式的组合。在 Bypass-Continuous 模式下，FIFO 缓冲区会开始处于 Bypass 模式，在发生选定的中断（例如，唤醒、自由落体、点击……）后切换为 Continuous 模式。

请按照以下步骤配置 Bypass-Continuous 模式：

1. 按照 Section 5 中断生成和嵌入功能中说明配置所需的中断发生器（确保锁定）。
2. 将 FTH[7:0]置为 255。
3. 将 INT2_FTH 置为‘1’：使能 FIFO 阈值中断
4. 将 FIFO_CTRL 寄存器（25h）中的 FMODE[2:0]字段设为 100b 可激活 Bypass-Continuous 模式。

当产生 FTH 中断时，从加速度计输出寄存器读取数据。

图 16. 旁路-连续模式



如图 16. 旁路-连续模式中所示，FIFO 初始处于 Bypass 模式，因此无采样进入 FIFO 缓冲区。一旦发生事件（例如唤醒或自由落体事件），则 FIFO 切换到 Continuous 模式并开始以所配置的数据速率存储样本。当达到所编程的阈值时，FTH 中断变为高电平，应用处理器会尽快开始读取所有 FIFO 采样（256 * 6 字节），以免丢失数据。

如果 FIFO_OVR 标志被置位，那么一旦读取第一个 FIFO 组，该标志就会变为 0，为新数据创建空间。由于 FIFO 仍处于 Continuous 模式，因此，FIFO 最终会再次达到阈值，重复此情况。

最后，要么中断事件被清除，要么 FIFO 直接进入 Bypass 模式，然后停止采集数据。

6.6 从 FIFO 读取数据

当 FIFO 模式不是 Bypass 时，读取输出寄存器（28h 至 2Dh）会返回早先的 FIFO 样本集合。

读取输出寄存器时，其内容会移至 SPI/I²C 输出缓冲区。理想地，FIFO 插槽会向上移动一格，以便释放空间接收新的采样，并且输出寄存器载入 FIFO 缓冲器中存储的当前最旧的值。

通过从加速度计输出寄存器执行 256 次读取操作，可以重新取回整个 FIFO 内容。无论功耗模式如何，存储在 FIFO 中的数据大小始终是 14 位。每隔一个读取操作都会返回相同的最终值，直到 FIFO 缓冲区中有一个新的样本集可用。

为了提高应用的灵活性，可使用每种读取字节组合从 FIFO 重新获取数据（例如：1536 次单字节读取，256 次 6 字节读取，1 次 1536 字节的多字节读取等）。

建议以 1536 字节的多字节读取（6 个输出寄存器乘以 256 个插槽）来读取所有 FIFO 插槽。为了减少主机和从机之间的通信，可以通过将 CTRL2 寄存器的 IF_ADD_INC 位置为“1”，使器件的读取地址自动递增；当到达寄存器 0x2D 时，器件回滚到 0x28。

I²C 速度低于 SPI，它需要大约 29 个时钟脉冲才能开始通信（开始、从地址，寄存器地址+写入、重新启动、寄存器地址+读取），并且每个字节读取都需要额外 9 个时钟脉冲（总共为 83 个时钟脉冲）。因此，在使用标准 I²C 模式的情况下（最大速率为 100 kHz），单个样本集读取需要耗费

830 µs，总 FIFO 下载需要耗费约 138.53 ms（29 + 9 * 1536 时钟脉冲）。

在 SPI 的情况下，相反，只需要在开始启动时耗费 9 个时钟脉冲（r / w +寄存器地址），再加上额外的每个字节读取耗费 8 个时钟脉冲。使用 2 MHz 时钟时，单个采样集读取将需要 28.5µs，总 FIFO 下载大约需要 6.15 ms。

如果按照此建议，使用标准 I²C（100 kHz），那么全部读取 FIFO（138.53 ms）将需要耗费 14/ODR，ODR 为 100 Hz。使用 SPI @ 2 MHz（器件支持的最大速率是 10 MHz）时，全部读取 FIFO 将需要耗费 1/ODR，ODR 为 100 Hz。

因此，为了不丢失样本，应用将在 FIFO 满之前读取样本，设置阈值并使用 FTH 中断（参见章节 Section 6.3 FIFO 中断）。

表 32. 示例：ODR 阈值功能

ODR（Hz）	FTH_THS （I²C @ 100 kHz）	FTH_THS （I²C @ 400 kHz）	FTH_THS (SPI @ 2 MHz)
50	36	147	256
100	17	73	256
200	8	36	208
400	4	17	103
800	1	8	51
1600	-	4	25

7 温度传感器

LIS2DS12 具有内部温度传感器，适用于环境温度测量。

如果加速度计传感器处于掉电模式，则温度传感器关闭并显示最后的测量值。

温度传感器的输出数据速率固定为 12.5 Hz。

温度数据由 OUT_T 寄存器给出，以二进制补码的格式表示为一个 8 位的数字，其灵敏度为+1 LSB/°C。输出零值对应于 25 °C ±15 °C。

7.1 温度数据计算示例

表 33. 输出数据寄存器内容 vs. 温度 提供了在不同环境温度值下从温度数据寄存器中读取数据的几个基本示例。本表中所列值是在理想器件校准的假设下给出的（即，无偏移，无增益误差，……）。

表 33. 输出数据寄存器内容 vs. 温度

温度值	OUT_T (26h)
23 °C	FEh
24 °C	FFh
25 °C	00h
27 °C	02h

8 自检功能

嵌入式自检功能可支持无需移动器件而对其功能进行检查。

8.1 加速度计自检

当启用加速度计自检时，致动力会施加到传感器，致使传感器的可移动部分发生挠曲，并产生加速度。这种情况下，传感器输出会在其 DC 电平上表现出变化，该电平通过灵敏度值关联到所选满量程。

当 CTRL3 寄存器的 ST[2:1]位被设定为 00b 时，加速度计自检功能关闭；当 ST[2:1]位被置为 01b（正符号自检）或 10b（负符号自检）时，该功能使能。

当加速度计自检功能激活时，传感器输出电平由作用在传感器上的加速度和静电测试力的代数和给出。

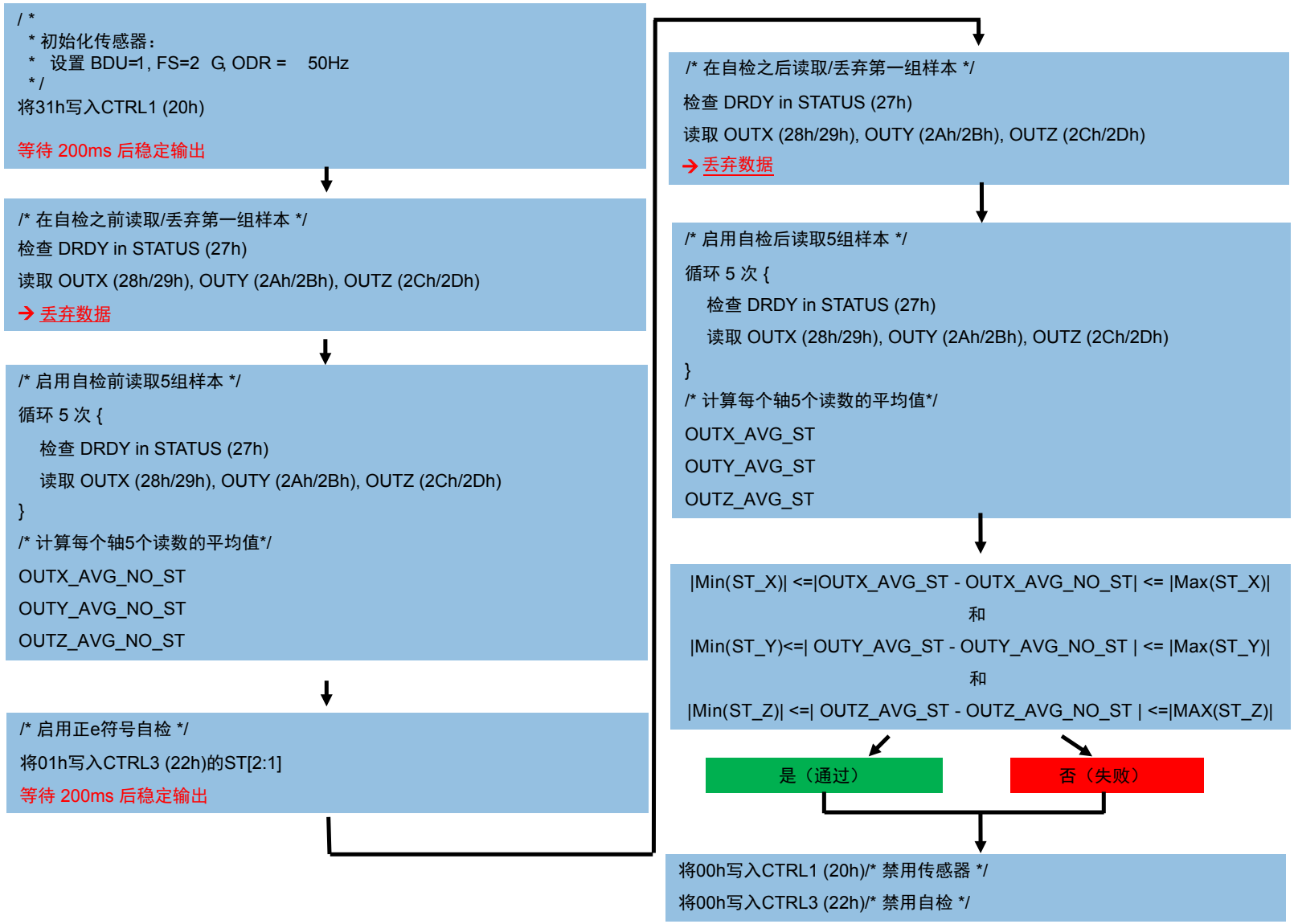
该过程包括：

1. 使能加速度计
2. 启动自检之前，对五个采样取平均
3. 启动自检之后，对五个采样取平均
4. 计算每个轴的模差，并验证它是否落在给定的范围内。数据表中给出了最小和最大值。

完整的加速度计自检过程如图 17. 加速度计自检步骤中所示。

注意：自检过程中保持器件静止。

图 17. 加速度计自检步骤



版本历史

表 34. 文档版本历史

日期	版本	变更
2015 年 11 月 5 日	1	初始版本
2017 年 5 月 10 日	2	更新了 1 节“前言”， 表 1：“寄存器”， 第 3.4 节：“加速度计带宽”和图 1：“加速度计采样链路图”， 3.4.1 节：“加速度计斜率滤波器”， 第 4.3 节：“使用数据准备就绪信号”和图 3：“数据就绪”信号， 4.4 节：“使用块数据更新（block data update，BDU）功能”， 4.5 节：“认识输出数据”， 4.5.1 节：“输出数据示例”， 第 5 节：“中断生成和嵌入功能”， 第 5.1 节：“中断引脚配置”， 5.3 节：“自由落体中断”， 5.4 节：“唤醒中断”， 第 5.5.1 节：“6D 方向”和图 6：“6D 识别方向”， 第 5.6.2 节：“双击”和图 8：“双击事件识别（LIR 位= 0）， 5.6.3 节：“单击和双击识别配置， 5.6.4 节：“单击示例”， 5.6.5 节：“双击示例”， 第 5.7 节：“活动/不活动识别”和图 10：“活动/不活动识别”， 第 5.9 节：“嵌入功能”， 5.9.1 节：“计步功能：步伐侦测和步数计算”， 第 5.9.2 节：“大幅运动检测”， 第 5.10.1 节：“传感器集线器引脚说明”， 第 5.10.5 节：“传感器集线器示例”， 第 6.2.1 节：“FIFO_CTRL (25h)， 第 6.2.3 节：“FIFO_SRC (2Fh)， 第 6.2.4 节：“FIFO_SAMPLES (30h)， 第 6.3 节：“FIFO 中断”， 第 6.3.2 节：“FIFO 满”， 第 6.5.2 节：“FIFO 模式”和图 12：“FIFO 模式行为”， 6.5.3 节：“连续模式”， 第 6.5.4 节：“连续-FIFO 模式”和图 14：“连续-FIFO 模式：中断锁存和非锁存， 6.5.5：“旁路-连续模式” 6.6 节：“从 FIFO 中取数据”， 和第 7 节：“温度传感器”
2018 年 1 月 12 日	3	添加了第 2 节：“引脚说明” 更新了图 2：“加速度计采样链” 更新了图 4：“数据就绪信号” 更新了图 11：“活动/不活动识别”

日期	版本	变更
2018 年 9 月 6 日	4	更新了 Section 5.9.1 计步功能：步伐侦测和步数计算 更新了 Section 5.9.2 大幅运动检测 更新了 Section 6.4 模至 FIFO

目录

表一览

图一览