LIS2DW12：始终开启的 3D 加速度计

引言

本文档旨在提供 ST 的 LIS2DW12 运动传感器相关的使用信息和应用提示。

LIS2DW12 是系统级封装的 3D 数字加速度计，具有数字 I²C/SPI 串口标准输出，在高分辨率模式下功耗 90 µA，在低功耗模式下功耗低于 1 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* 的动态用户可选满量程加速度范围，并能通过 1.6 Hz 到 1600 Hz 的输出数据速率测量加速度。经过配置，LIS2DW12 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。

LIS2DW12 内置 32 级先进先出（FIFO）缓冲器提供给用户存储数据，可以减少主控的干预。

LIS2DW12 采用纤薄的小型塑料平面网格阵列封装(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

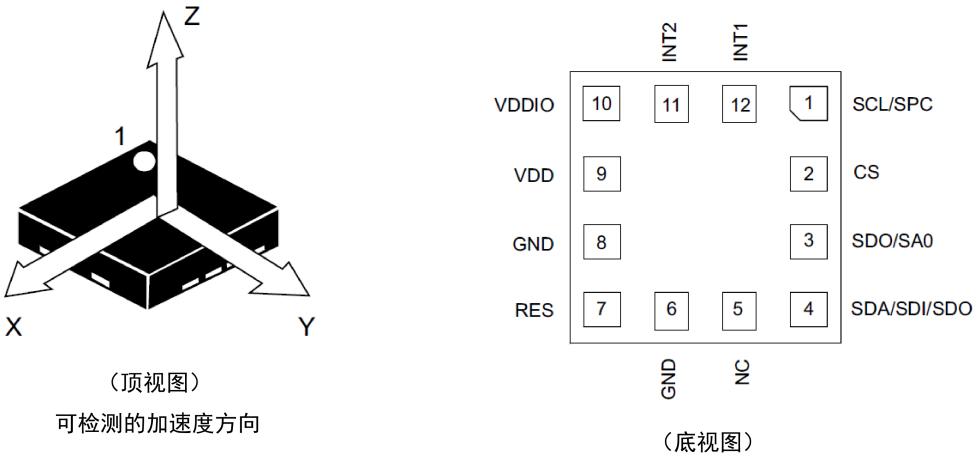


表 1. 内部引脚状态

引脚#	名称	功能	引脚状态
1	SCL SPC	I²C 串行时钟（SCL） SPI 串口时钟（serial port clock，SPC）	默认：开漏
2	CS	SPI 使能 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	默认：使用内部上拉输入(1)
3	SDO SA0	串行数据输出（SDO） I²C 设备地址的最低有效位（SA0）	默认：使用内部上拉输入(2)
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）	默认：（SDA）输入开漏
5	NC	内部未连接。可连接到 VDD、VDDIO 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	INT2	中断引脚 2。根据需要在单个数据转换中进行选择时的时钟输入。	默认值：推挽输出强制接地
12	INT1	中断引脚 1	默认值：推挽输出强制接地

1. *为禁止 CS 引脚上的内部上拉，将“1”写入 CTRL2（21h）的 CS_PU_DISC 位。*
2. *无法禁用 SDO/SA0 引脚上的内部上拉：在低功耗应用中，请勿将此引脚连接到 GND。*

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
OUT_T_L(1)	0Dh	TEMP3	TEMP2	TEMP1	TEMP0	0	0	0	0
OUT_T_H(1)	0Eh	TEMP11	TEMP10	TEMP9	TEMP8	TEMP7	TEMP6	TEMP5	TEMP4
WHO_AM_I(1)	0Fh	0	1	0	0	0	1	0	0
CTRL1	20h	ODR3	ODR2	ODR1	ODR0	MODE1	MODE0	LP_MODE1	LP_MODE0
CTRL2	21h	BOOT	SOFT_RESET	0	CS_PU_DISC	BDU	IF_ADD_INC	I2C_DISABLE	SIM
CTRL3	22h	ST2	ST1	PP_OD	LIR	H_LACTIVE	0	SLP_MODE _SEL	SLP_MODE_1
CTRL4_INT1_PAD_CTRL	23h	INT1_6D	INT1_ SINGLE_TAP	INT1_WU	INT1_FF	INT1_TAP	INT1_DIFF5	INT1_FTH	INT1_DRDY
CTRL5_INT2_PAD_CTRL	24h	INT2_ SLEEP_ 状态	INT2_ SLEEP_CHG	INT2_BOOT	INT2_ DRDY_T	INT2_OVR	INT2_DIFF5	INT2_FTH	INT2_DRDY
CTRL6	25h	BW_FILT1	BW_FILT0	FS1	FS0	FDS	LOW_NOISE	0	0
OUT_T(1)	26h	TEMP7	TEMP6	TEMP5	TEMP4	TEMP3	TEMP2	TEMP1	TEMP0
STATUS(1)	27h	FIFO_THS	WU_IA	SLEEP_STATE	DOUBLE_TAP	SINGLE_TAP	6D_IA	FF_IA	DRDY
OUT_X_L(1)	28h	X_L7	X_L6	X_L5	X_L4	X_L3(2)	X_L2(2)	0	0
OUT_X_H(1)	29h	X_H7	X_H6	X_H5	X_H4	X_H3	X_H2	X_H1	X_H0
OUT_Y_L(1)	2Ah	Y_L7	Y_L6	Y_L5	Y_L4	Y_L3(2)	Y_L2(2)	0	0
OUT_Y_H(1)	2Bh	Y_H7	Y_H6	Y_H5	Y_H4	Y_H3	Y_H2	Y_H1	Y_H0
OUT_Z_L(1)	2Ch	Z_L7	Z_L6	Z_L5	Z_L4	Z_L3(2)	Z_L2(2)	0	0
OUT_Z_H(1)	2Dh	Z_H7	Z_H6	Z_H5	Z_H4	Z_H3	Z_H2	Z_H1	Z_H0
FIFO_CTRL	2Eh	FMode2	FMode1	FMode0	FTH4	FTH3	FTH2	FTH1	FTH0
FIFO_SAMPLES(1)	2Fh	FIFO_FTH	FIFO_OVR	Diff5	Diff4	Diff3	Diff2	Diff1	Diff0
TAP_THS_X	30h	4D_EN	6D_THS1	6D_THS0	TAP_THSX_4	TAP_THSX_3	TAP_THSX_2	TAP_THSX_1	TAP_THSX_0
TAP_THS_Y	31h	TAP_PRIOR_2	TAP_PRIOR_1	TAP_PRIOR_0	TAP_THSY_4	TAP_THSY_3	TAP_THSY_2	TAP_THSY_1	TAP_THSY_0
TAP_THS_Z	32h	TAP_X_EN	TAP_Y_EN	TAP_Z_EN	TAP_THSZ_4	TAP_THSZ_3	TAP_THSZ_2	TAP_THSZ_1	TAP_THSZ_0
INT_DUR	33h	LATENCY3	LATENCY2	LATENCY1	LATENCY0	QUIET1	QUIET0	SHOCK1	SHOCK0
WAKE_UP_THS	34h	SINGLE_ DOUBLE_TAP	SLEEP_ON	WK_THS5	WK_THS4	WK_THS3	WK_THS 2	WK_THS 1	WK_THS 0

寄存器名	地址	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
WAKE_UP_DUR	35h	FF_DUR5	WAKE_DUR1	WAKE_DUR0	STATIONARY	SLEEP_DUR3	SLEEP_DUR2	SLEEP_DUR1	SLEEP_DUR0
FREE_FALL	36h	FF_DUR4	FF_DUR3	FF_DUR2	FF_DUR1	FF_DUR0	FF_THS2	FF_THS1	FF_THS0
STATUS_DUP(1)	37h	OVR	DRDY_T	SLEEP_STATE_IA	DOUBLE_TAP	SINGLE_TAP	6D_IA	FF_IA	DRDY
WAKE_UP_SRC(1)	38h	0	0	FF_IA	SLEEP_STATE IA	WU_IA	X_WU	Y_WU	Z_WU
TAP_SRC(1)	39h	0	TAP_IA	SINGLE_ TAP	DOUBLE_ TAP	TAP_SIGN	X_TAP	Y_TAP	Z_TAP
SIXD_SRC(1)	3Ah	0	6D_IA	ZH	ZL	YH	YL	XH	XL
ALL_INT_SRC(1)	3Bh	0	0	SLEEP_ CHANGE_IA	6D_IA	DOUBLE_TAP	SINGLE_TAP	WU_IA	FF_IA
X_OFS_USR	3Ch	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	3Dh	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	3Eh	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
CTRL7	3Fh	DRDY_ PULSED	INT2_ON_INT1	中断 _ENABLE	USR_OFF_ON _OUT	USR_OFF_ON _WU	USR_OFF_W	HP_REF_ MODE	LPASS_ON6D

1. *只读寄存器*
2. *如果使能低功耗模式 1，则该位置 0。*

3 工作模式

3.1 功率模式

设计了五组工作模式，为客户提供广泛的噪声/功耗组合选择：

- 1 高性能模式：注重低噪音
- 4 低功耗模式：噪音和功耗取得折中

表 3. 加速度计分辨率

高性能模式	低功耗模式 4	低功耗模式 3	低功耗模式 2	低功耗模式 1
14 位	14 位	14 位	14 位	12 位

通过写入下表所示的 CTRL1（20h）中的 MODE[1:0]和 LP_MODE[1:0]位来选择这些工作模式。

表 4. CTRL1 寄存器

b7	b6	b5	b4	b3	b2	b1	b0
ODR3	ODR2	ODR1	ODR0	MODE1	MODE0	LP_MODE1	LP_MODE0

表 5. 模式选择

MODE[1:0]	模式和分辨率
00	低功耗模式（12/14 位分辨率）
01	高性能模式（14 位分辨率）
10	按需模式下的单个数据转换（12/14 位分辨率）
11	不允许

表 6. 低功耗模式选择

LP_MODE[1:0]	模式和分辨率
00	低功耗模式 1（12 位分辨率）
01	低功耗模式 2（14 位分辨率）
10	低功耗模式 3（14 位分辨率）
11	低功耗模式 4（14 位分辨率）

根据上述五组工作模式的每一组，设计了两种配置：

- 极低功耗（低噪音关闭）
- 低噪音

在 CTRL6（25h）中写入 LOW_NOISE 位可选择所需的配置。CTRL6（25h）中的 LOW_NOISE 位会影响前端噪音和电流消耗。带宽和建立时间不受影响。

表 7. **1.8 V 时的功耗 [uA]**

输出数据频率	高性能		低功耗模式 4		低功耗模式 3		低功耗模式 2		低功耗模式 1	
	LOW_NOISE		LOW_NOISE		LOW_NOISE		LOW_NOISE		LOW_NOISE	
	0	1	0	1	0	1	0	1	0	1
1.6 Hz	-	-	0.65	0.7	0.55	0.6	0.45	0.5	0.38	0.4
12.5 Hz	90	120	4	5	2.5	3	1.6	2	1	1.1
25 Hz	90	120	8.5	10	4.5	6	3	3.5	1.5	2
50 Hz	90	120	16	20	9	11	5.5	7	3	3.5
100 Hz	90	120	32	39	17.5	21.5	10.5	13	5	6
200 Hz	90	120	63	77	34.5	42	20.5	25	10	12
400/800/1600 Hz	90	120	-	-	-	-	-	-	-	-

表 8. **ODR = 200 Hz 时的噪音[μg/√Hz]**

满量程	高性能		低功耗模式 4		低功耗模式 3		低功耗模式 2		低功耗模式 1	
	LOW_NOISE		LOW_NOISE		LOW_NOISE		LOW_NOISE		LOW_NOISE	
	0	1	0	1	0	1	0	1	0	1
2 *g*	110	90	160	130	210	180	300	240	550	450
4 *g*	110	100	170	140	230	190	320	270	650	540
8 *g*	130	120	170	150	240	210	330	280	680	580
16 *g*	170	160	200	180	270	240	370	330	770	700

3.2 连续转换（Continuous conversion）

当 CTRL1（20h）中的 MODE[1:0]位设置为低功耗[00]或高性能模式[01]时，设备处于连续转换状态，输出数据速率可通过 CTRL1（20h）中的 ODR[3:0]位进行选择。

表 9. **输出数据率选择**

ODR[3:0]	模式和分辨率
0000	下电
0001	高性能 12.5 Hz / 低功耗模式 1.6 Hz
0010	12.5 Hz（与功耗模式无关）
0011	25 Hz（与功耗模式无关）
0100	50 Hz（与功耗模式无关）
0101	100 Hz（与功耗模式无关）
0110	200 Hz（与功耗模式无关）
0111	高性能 400 Hz / 低功耗模式 200 Hz
1000	高性能 800 Hz / 低功耗模式 200 Hz
1001	高性能 1600 Hz / 低功耗模式 200 Hz

3.3 单个数据转换（按需模式）

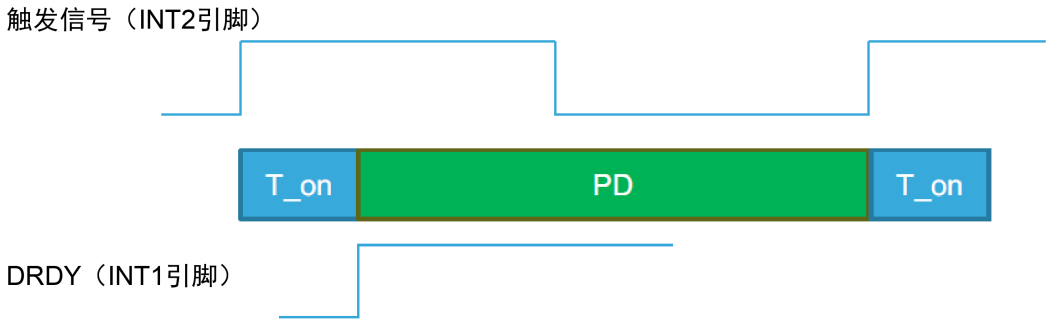
此模式仅适用于低功耗模式，可通过在 CTRL1（20h）中将 MODE [1:0]位写入“10”来使能。

在此配置中，设备等待触发信号，以便根据 CTRL1（20h）中所选的功耗模式 LP_MODE [1:0]位生成新数据，之后设备立即进入掉电状态。

触发器可以是：

- INT2 引脚的上升沿（如果寄存器 CTRL3（22h）中的 SLP_MODE_SEL ='0'）。此时，用户可以使用 STATUS 寄存器（27h）的 DRDY 位检测转换何时结束，通过将寄存器 CTRL4_INT1_PAD_CTRL（23h）中的 INT1_DRDY 位置为 1，也可以将其发送至 INT1 引脚。触发信号高电平的最小持续时间为 20 ns。
- 在寄存器 CTRL3（22h）中将 SLP_MODE_1 写入’1’（如果寄存器 CTRL3（22h）中的 SLP_MODE_SEL =’1’）。此时，如前一种情况所示，用户可以使用 DRDY 位/信号检测转换何时结束，或者检查寄存器 CTRL3（22h）中的 SLP_MODE_1 位何时自动清零。

图 2. 使用 INT2 作为外部触发器进行单个数据转换（SLP_MODE_SEL = 0）



使用单个数据转换模式的最大数据速率为 200 Hz，转换时间取决于所选的低功耗模式（参见下表）。

表 10. 低功耗模式选择

低功耗	典型转换时间 (T_on)
模式 1	1.20 ms
模式 2	1.70 ms
模式 3	2.30 ms
模式 4	3.55 ms

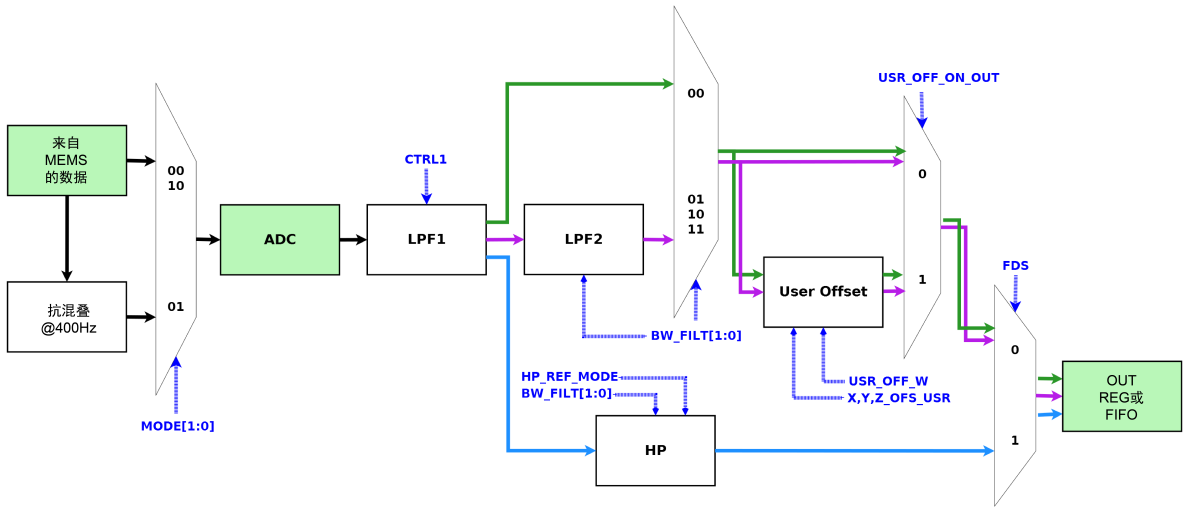
使用单个数据转换模式时，仍支持中断、嵌入式功能和 FIFO。同时，嵌入式滤波器 LPF1、LPF2 和 HP 也可用于单个数据转换（按需模式），其带宽和建立时间与所选低功耗模式相同（详细信息，请参见 Section 3.4 加速度计带宽）。

3.4 加速度计带宽

加速度计采样链（图 3. 加速度计滤波链图） 由几个级联模块表示：

- ADC：模数转换器
- 抗混叠滤波器：仅在高性能模式（MODE [1:0] = 01）中可用，截止频率为 400 Hz
- LPF1(2)：低通滤波器 1(2)
- HP：高通滤波器
- 用户偏移：从采样数据中减去的可配置值（每个轴一个）

图 3. 加速度计滤波链图



如上图所示，可以使用三种不同的滤波器路径生成数据：

- 仅 LPF1（绿色路径）：为了选择此路径，设置 BW_FILT [1:0] = 00。其他详细信息请参见 表 11. 低通滤波器 1 带宽。
- LPF1 + LPF2（紫色路径）：为了选择此路径，设置 BW_FILT[1:0]的值不同于 00。其他详细信息请参见 表 12. 带宽：高通和低通路径。
- LPF1 + HP（蓝色路径）：这些输出始终可用于 BW_FILT [1:0] = 00 的每个值。其他详细信息请参见 表 12. 带宽：高通和低通路径。

表 11. 低通滤波器 1 带宽

模式	ODR 选择	BW_FILT[1:0] = 00	
		待丢弃的样本(1) 建立@95%	截止[Hz]
低功耗模式 4	@ 每个 ODR	0	180
低功耗模式 3	@ 每个 ODR	0	360
低功耗模式 2	@ 每个 ODR	0	720
低功耗模式 1	@ 每个 ODR	0	3200
高性能	@12.5 Hz	0	ODR/2
高性能	@25 Hz	0	ODR/2
高性能	@50 Hz	0	ODR/2
高性能	@100 Hz	1	ODR/2
高性能	@200 Hz	1	ODR/2

模式	ODR 选择	BW_FILT[1:0] = 00	
		待丢弃的样本(1) 建立@95%	截止[Hz]
高性能	@400 Hz	1	ODR/2
高性能	@800 Hz	1	ODR/2
高性能	@1600 Hz	2	400

1. *ODR [3:0]、MODE [1:0]、LP_MODE [1:0]和 BW_FILT [1:0]的起始条件不会影响这些值。导通时间（从掉电条件开始可用的第一个样品）为 1 / ODR。*

表 12. 带宽：高通和低通路径

模式	ODR 选择	BW_FILT[1:0] = 01		BW_FILT[1:0] = 10		BW_FILT[1:0] = 11	
		待丢弃的样本(1) 建立@95%	截止[Hz]	待丢弃样本(1) 建立@95%	截止 [Hz]	待丢弃样本(1) 建立@95%	截止 [Hz]
LP 模式 4	@ 每个 ODR	1	ODR/4	5	ODR/10	11	ODR/20
LP 模式 3	@ 每个 ODR	1	ODR/4	5	ODR/10	11	ODR/20
LP 模式 2	@ 每个 ODR	1	ODR/4	5	ODR/10	11	ODR/20
LP 模式 1	@ 每个 ODR	1	ODR/4	5	ODR/10	11	ODR/20
HP	@12.5 Hz	1	ODR/4	5	ODR/10	11	ODR/20
HP	@25 Hz	1	ODR/4	5	ODR/10	11	ODR/20
HP	@50 Hz	1	ODR/4	5	ODR/10	11	ODR/20
HP	@100 Hz	1	ODR/4	5	ODR/10	11	ODR/20
HP	@200 Hz	2	ODR/4	5	ODR/10	11	ODR/20
HP	@400 Hz	2	ODR/4	5	ODR/10	11	ODR/20
HP	@800 Hz	2	ODR/4	5	ODR/10	11	ODR/20
HP	@1600 Hz	3	ODR/4	6	ODR/10	12	ODR/20

1. *ODR [3:0]、MODE [1:0]、LP_MODE [1:0]和 BW_FILT [1:0]的起始条件不会影响这些值。*

在 CTRL7 中设置 USR_OFF_ON_OUT = 1 不会改变系统的带宽。在该配置中，从各个轴中减去写入寄存器 X_OFS_USR、Y_OFS_USR、Z_OFS_USR 中的值。偏移值带有符号（二进制补码）。

寄存器 X_OFS_USR、Y_OFS_USR、Z_OFS_USR 中各个位的权重是通过 CTRL7 中的 USR_OFF_W 位定义的。

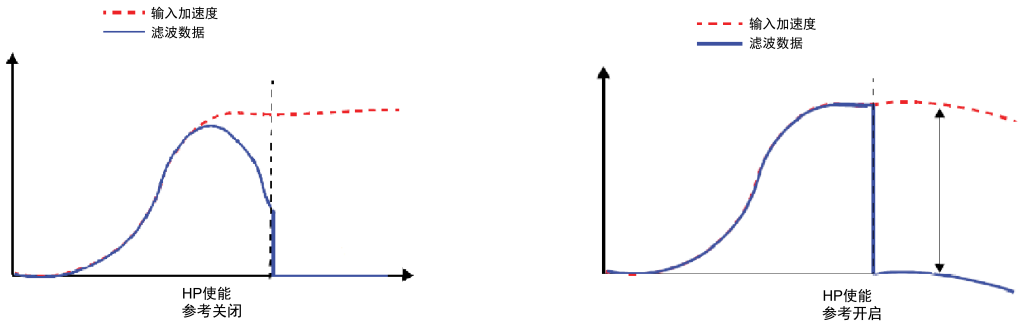
3.5 高通滤波器配置

LIS2DW12 提供的嵌入式高通滤波功能可轻松去除测得加速度的 DC 分量。如图 3. 加速度计滤波链图所示，通过寄存器 CTRL6 中的 FDS 位，用户可以将滤波器输出发送至输出寄存器。

也可以将滤波器独立应用于嵌入式功能数据（见 Section 5 中断生成和嵌入功能的图 6. 嵌入功能）。这意味着可以在中断生成作用于未滤波数据的同时获得已进行滤波的数据。

可以在*参考模式*设置高通滤波器。在该配置下，输出数据会计算为输入加速度与使能*参考模式*时收集的数值之差。这样一来，仅应用差值，无需滤波。

图 4. 正常和参考模式下的高通滤波器



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。

请注意：VDD 不能低于 VDDIO。VDD = 0 V 且允许 VDDIO“常供”。

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。

有关数据生成的详细说明，请参见 Section 3 工作模式。

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。

应当按照如下步骤进行读取操作：

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

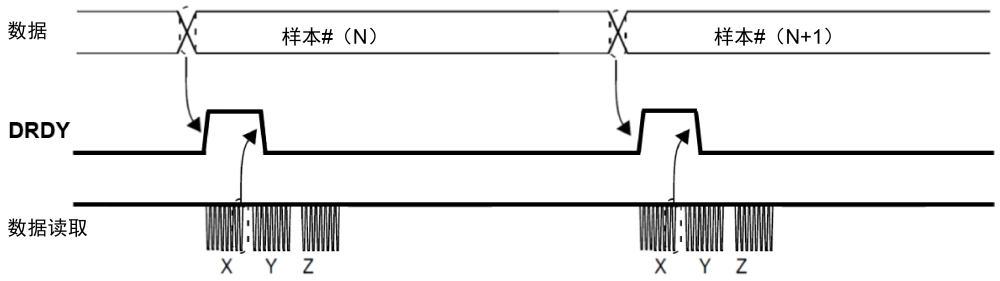
4.3 使用数据准备就绪信号

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。

数据就绪信号源自于 STATUS 寄存器的 DRDY 位。通过将 CTRL4_INT1_PAD_CTRL 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_INT2_PAD_CTRL 寄存器的 INT2_DRDY 位置为 1，将其驱动至 INT2 引脚。

当一组新数据测量完毕并可读取时，数据准备就绪信号升高为 1。在 DRDY 锁存模式（CTRL7 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 5. 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 CTRL2（21h）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，但如果发起了对给定寄存器对（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。

加速度数据表示为二进制补码格式的 16 位数字，左对齐。根据所选的工作模式，这些值（LSB）具有不同的分辨率。

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。

表 13. 灵敏度

满量程	灵敏度[mg/LSB]	
	12 位格式(1)	14 位格式
±2 *g*	0.976	0.244
±4 *g*	1.952	0.488
±8 *g*	3.904	0.976
±16 *g*	7.808	1.952

1. *仅低功耗模式 1*

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，无增益误差等等）。
从传感器获取原始数据（高性能模式，±2 *g*）：

```
OUTX_L: 60h
```

```
OUTX_H: FDh
```

```
OUTY_L: 78h
```

```
OUTY_H: 00h
```

```
OUTZ_L: FCh
```

```
OUTZ_H: 42h
```

将寄存器串联：

```
OUTX_H & OUTX_L: FD60h
```

```
OUTY_H & OUTY_L: 0078h
```

```
OUTZ_H & OUTZ_L: 42FCh
```

应用灵敏度（例如，14 位分辨率，满量程为±2 *g* 时，使用 0.244）：

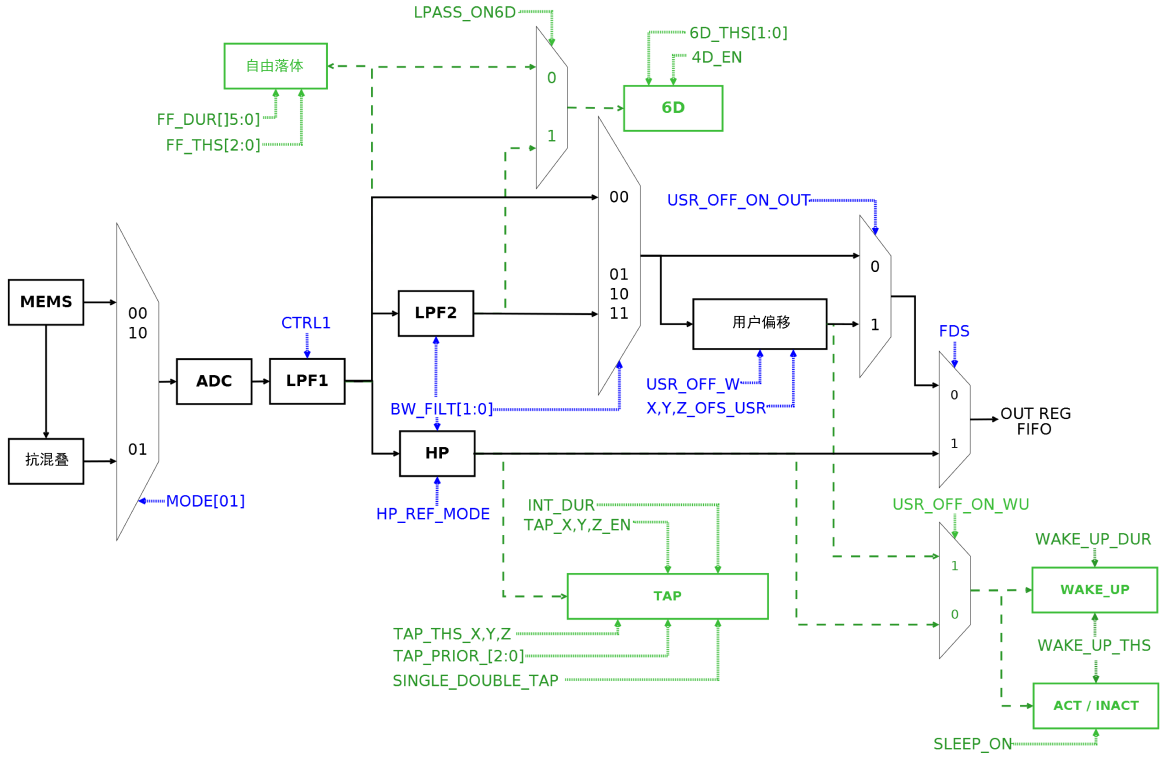
```
X: -672 / 4 * 0.244 = -41 mg
```

```
Y: +120 / 4 * 0.244 = +7 mg
```

```
Z: +17148 / 4 * 0.244 = +1046 mg
```

5 中断生成和嵌入功能

图 6. 嵌入功能



如要产生中断，必须将 LIS2DW12 设备设置为活动工作模式（不能处于掉电模式），因为中断的产生基于加速度计数据。

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。

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，可被驱动至 INT1 和/或 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。

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。

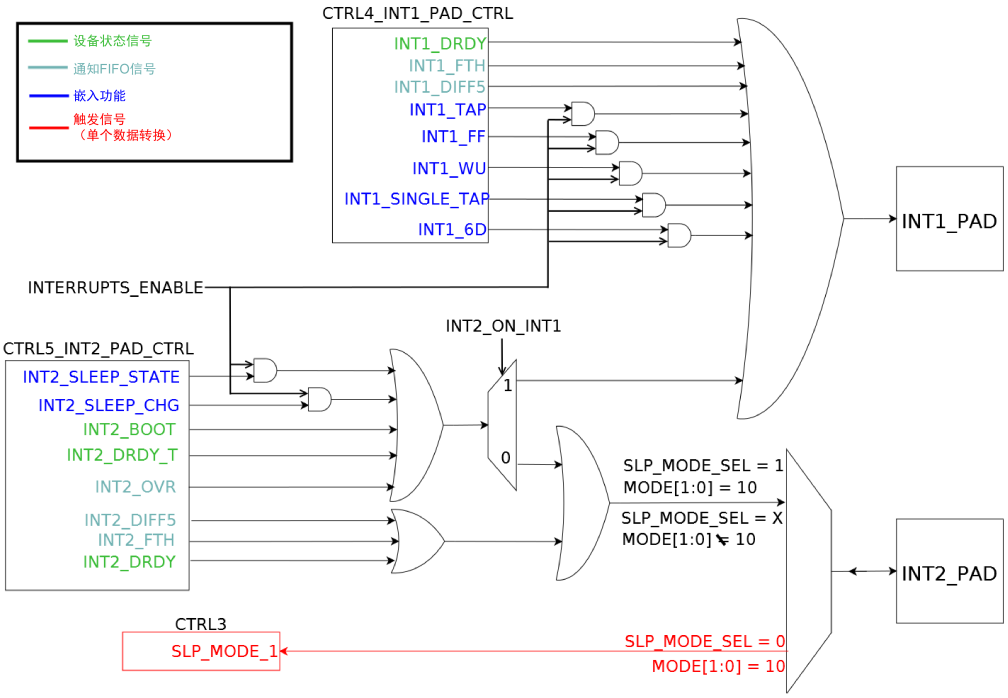
5.1 中断引脚配置

该设备具有两个引脚，可激活引脚来生成：

- *设备状态信号*（例如数据就绪和启动）
- *嵌入功能中断信号*
- *通知 FIFO 信号。*

所有*嵌入功能中断信号*均从属于寄存器 CTRL7 中的 INTERRUPTS_ENABLE 位。如果已经设置了该位，则在引脚上发送中断，否则仅通过读取其相对状态或源寄存器才可以使用这些中断。当 INT2 引脚在*单个数据转换（按需模式）*下用作外部触发器时，它也可以成为输入引脚。为了在此模式下配置设备，用户必须在寄存器 CTRL1 中设置 MODE [1:0] = 10 位，并在寄存器 CTRL3 中设置 SLP_MODE_SEL = 0 位。通过在寄存器 CTRL7 中设置 INT2_ON_INT1 = 1，可以在 INT1 引脚上发送所有 INT2 引脚信号。

图 7. 中断引脚配置



以下是这些中断控制寄存器的说明：这些位的默认值等于 0，对应于‘禁用’。要使能引脚上特定中断信号的线路，须将相应位置为 1。

表 14. CTRL4_INT1_PAD_CTRL

b7	b6	b5	b4	b3	b2	b1	b0
INT1_6D	INT1_ SINGLE _TAP	INT1_WU	INT1_ FF	INT1_ TAP	INT1_ DIFF5	INT1_ FTH	INT1_ DRDY

- INT1_6D：6D 识别被发送到 INT1 引脚。
- INT1_SINGLE_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：双击事件识别被发送到 INT1 引脚。
- INT1_DIFF5：FIFO 完全识别被发送到 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 15. **CTRL5_INT2_PAD_CTRL**

b7	b6	b5	b4	b3	b2	b1	b0
INT2_ SLEEP_ 状态	INT2_ SLEEP_ CHG	INT2_ BOOT	INT2_ DRDY_T	INT2_ OVR	INT2_ DIFF5	INT2_ FTH	INT2_ DRDY

- INT2_SLEEP_STATE：使能发送 SLEEP_STATE 到 INT2 引脚。
- INT2_SLEEP_CHG：睡眠变化状态被发送到 INT2 引脚。
- INT2 _BOOT：启动状态发送到 INT2 引脚。
- INT2_DRDY_T：温度数据就绪信号被发送到 INT2 引脚。
- INT2 _OVR：FIFO 上溢中断被发送到 INT2 引脚。
- INT2_DIFF5：FIFO 完全识别被发送到 INT2 引脚。
- INT2_FTH：FIFO 阈值事件被发送到 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。

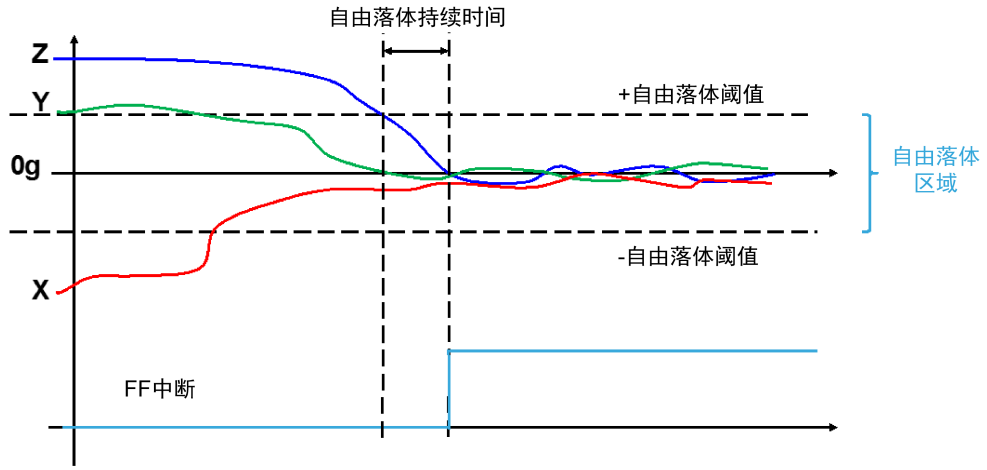
如下所示，STATUS 寄存器在地址 37h 处被复制，以便对连续寄存器进行多次读取。

- STATUS（27h）或 STATUS_DUP（37h）
- WAKE_UP_SRC (38h)
- TAP_SRC (39h)
- SIXD_SRC (3Ah)
- ALL_INT_SRC (3Bh)

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 8. 自由落体中断）。

图 8. 自由落体中断



通过将 CTRL4_INT1_PAD_CTRL 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 STATUS 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 或 ALL_INT_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，STATUS 中的 FF_IA 位复位）。

可以通过配置 FREE_FALL 寄存器（包含 FF_THS [2:0]位和 FF_DUR [4:0]位）和 WAKE_UP_DUR（包含持续时间参数的 MSB - FF_DUR5）寄存器来修改中断。LSB 的阈值请参见表 16. 自由落体阈值所述，并以 31.25mg 为单位表示。此表中给出的 LSB 值对于所有加速度计满量程范围均相同。

表 16. 自由落体阈值

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

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 64h 写入 CTRL1	// 启动加速度计 // ODR = 200 Hz，高性能
2. 将 04h 写入 CTRL6	// 使能 FS ±2 *g* LOW_NOISE
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_INT1_PAD_CTRL	// FF 中断驱动至 INT1 引脚
5. 将 10h 写入 CTRL3	// 锁存中断
6. 将 20h 写入 CTRL7	// 使能中断

示例代码中利用设置为~310 m*g*（31.25 m*g* * 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]字段配置：忽略短于 6/ODR = 6/200 Hz = 30 ms 的事件，以避免错误检测。

5.4 唤醒中断

在 LIS2DW12 设备中，可以通过 CTRL7 中的 USR_OFF_ON_WU 位，帮助唤醒功能选择使用高通滤波器或偏移输出，如图 6. 嵌入功能所示。

如果选择“偏移输出”，则每个轴都可以具有不同的偏移值，并写入寄存器 X_OFS_USR、Y_OFS_USR、Z_OFS_USR。位权重由寄存器 CTRL7 中的 USR_OFF_W 位定义。

如果一定数量的连续数据超出了所配置阈值，则产生唤醒中断信号（图 9. 唤醒事件识别（利用 HP 滤波器））。

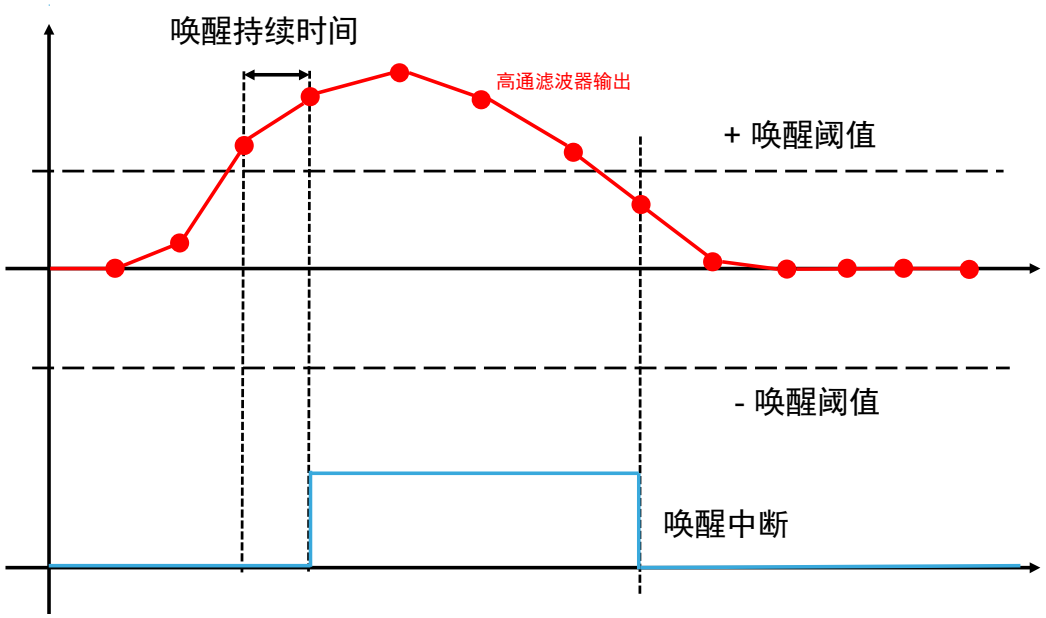
该无符号阈值由 WAKE_UP_THS 寄存器的 WK_THS [5:0]位来定义；这些 6 位的 1 LSB 值取决于所选加速度计满量程：1 LSB = FS/64。阈值可应用于正负数据：对于唤醒中断生成，三个轴中至少有一个必须大于阈值。

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 WAKE_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_INT1_PAD_CTRL 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 STATUS 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 或 ALL_INT_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（当不再检验到唤醒条件时，WAKE_UP_SRC 或 ALL_INT_SRC 中的 WU_IA 位复位）。

图 9. 唤醒事件识别（利用 HP 滤波器）



下面给出的示例代码实现了利用 HP 滤波器进行唤醒事件识别的 SW 程序。

1.	将 64h 写入 CTRL1	// 启动加速度计 // ODR = 200 Hz，高性能
2.	将 04 写入 CTRL6	// 使能 FS ±2 *g* LOW_NOISE
3.	将 20h 写入 CTRL7	// 使用 HP 滤波器，使能中断
4.	将 00h 写入 WAKE_UP_DUR	// 无持续
5.	将 02h 写入 WAKE_UP_THS	// 设置唤醒阈值
6.	将 20h 写入 CTRL4_INT1_PAD_CTRL	// 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 的 HP 滤波器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）。

下面给出的示例代码实现了利用 USER OFFSET 识别来执行唤醒事件的 SW 程序。

1.	将 04h 写入 CTRL6	// 使能 FS ±2 *g* LOW_NOISE // 使用 X/Y/Z_OFS_USR 寄存器
2.	将 34h 写入 CTRL7	// X/Y/Z_OFS_USR 权重 15.6 mg/LSb // 使能中断
3.	将 00h 写入 X_OFS_USR	// 将 X 偏移设置为 0
4.	将 00h 写入 Y_OFS_USR	//将 Y 偏移设置为 0
5.	将 40h 写入 Z_OFS_USR	//将 Z 偏移设置为 1*g*
6.	将 00h 写入 WAKE_UP_DUR	// 无持续
7.	将 02h 写入 WAKE_UP_THS	// 设置唤醒阈值
8.	将 20h 写入 CTRL4_INT1_PAD_CTRL	// 唤醒中断驱动至 INT1 引脚
9.	将 64h 写入 CTRL1	// 启动加速度计 // ODR = 200 Hz，高性能

由于持续时间被设置为 0，因此，当测得的数据与_OFS_USR 寄存器之间的差值超过所配置阈值时，每个 X、Y、Z 数据便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

LIS2DW12 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：SIXD_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 17. SIXD_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加速度的绝对值大于阈值时，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加速度的绝对值大于阈值时，ZL（YL、XL）被置为高电平。

TAP_THS_X 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 18. 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 18. 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_INT1_PAD_CTRL 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 SIXD_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 SIXD_SRC 或 ALL_INT_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 10. 6D 识别方向中所示的六种可能情形，表 19. 用于 6D 定位的 SIXD_SRC 寄存器中显示了每个位置对应的 SIXD_SRC 寄存器内容所示。

图 10. 6D 识别方向

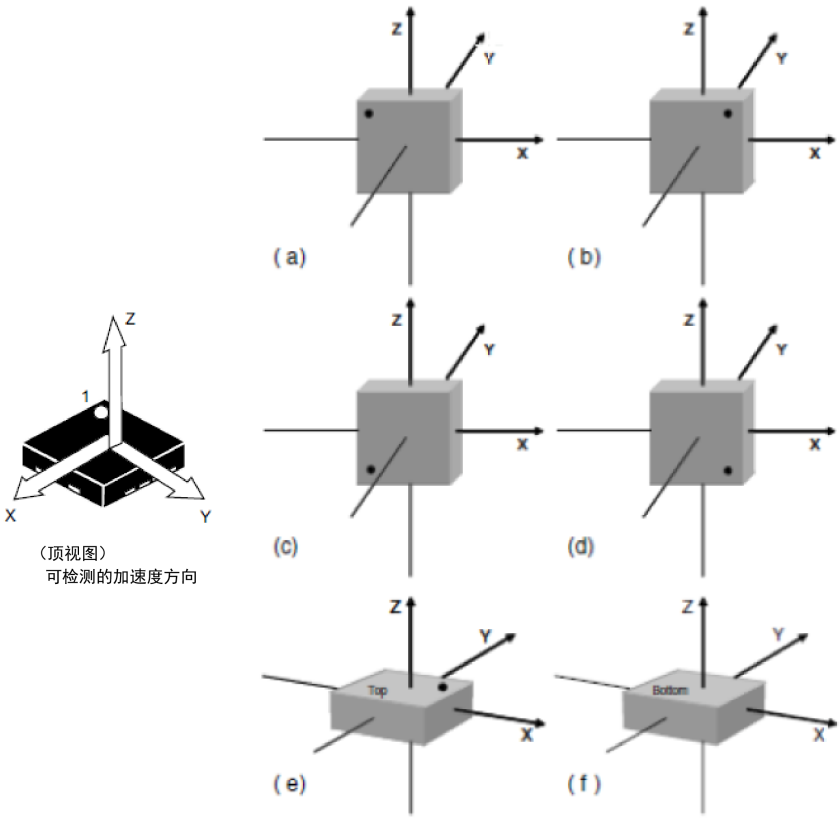


表 19. 用于 6D 定位的 SIXD_SRC 寄存器

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

下面的示例实现了用于 6D 检测的软件流程：

1. 将 64h 写入 CTRL1 // 启动加速度计 // ODR = 200 Hz，高性能
2. 将 04h 写入 CTRL6 // 使能 FS ±2 *g* LOW_NOISE
3. 将 20h 写入 CTRL7 // 请勿在 6D 中使用低通滤波器，使能中断
4. 将 40h 写入 TAP_THS_X // 设置 6D 阈值（6D_THS[1:0] = 10b = 60 degrees）
5. 将 80h 写入 CTRL4_INT1_PAD_CTRL // 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_THS_X 寄存器的 4D_EN 位置为 1 来使能。在该配置下，Z 轴位置检测会禁用，从而可将位置识别缩小为表 19. 用于 6D 定位的 SIXD_SRC 寄存器的用例(a)、(b)、(c)和(d)。

5.6 单击和双击识别

LIS2DW12 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。

在 LIS2DW12 设备中，单击和双击识别功能使用高通滤波器来检测单击和双击事件。

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高通滤波数据幅度和时序进行编程。

推荐在单击和双击识别中使用 400 Hz 或更高的加速度计 ODR。

5.6.1 单击

如果设备配置为单击事件检测，那么当高通滤波器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。

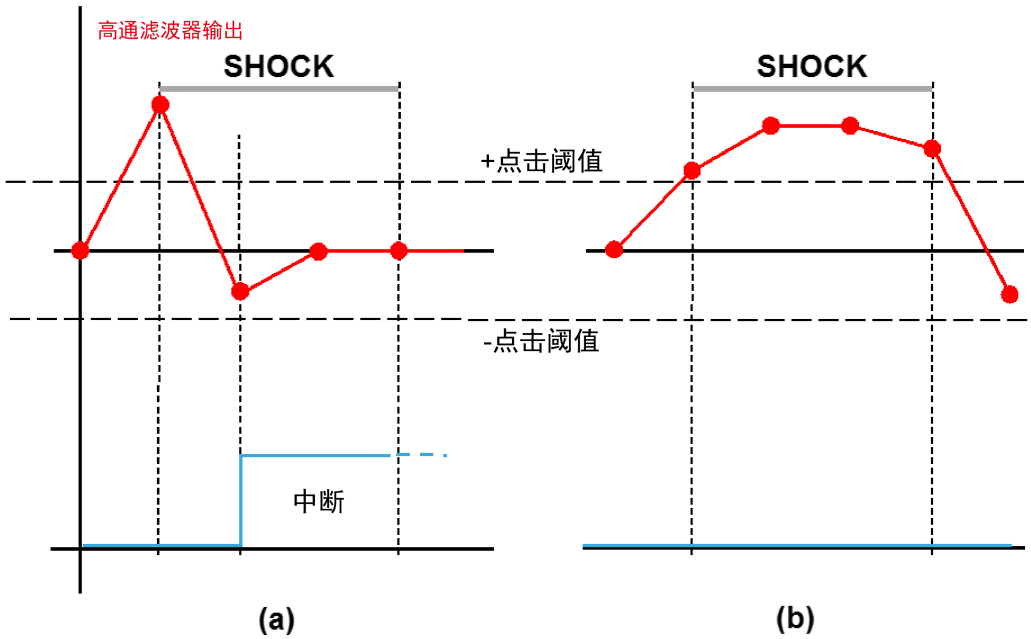
在单击情况下，如果 CTRL3 寄存器的 LIR 位被置为 0，则中断在 Quiet 窗口持续时间内保持高电平。

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 或 ALL_INT_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 11. 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 时间窗口之后信号数据低于阈值，因此未识别出点击 。

图 11. 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，高通滤波数据超过阈值时，才开始第二次点击识别过

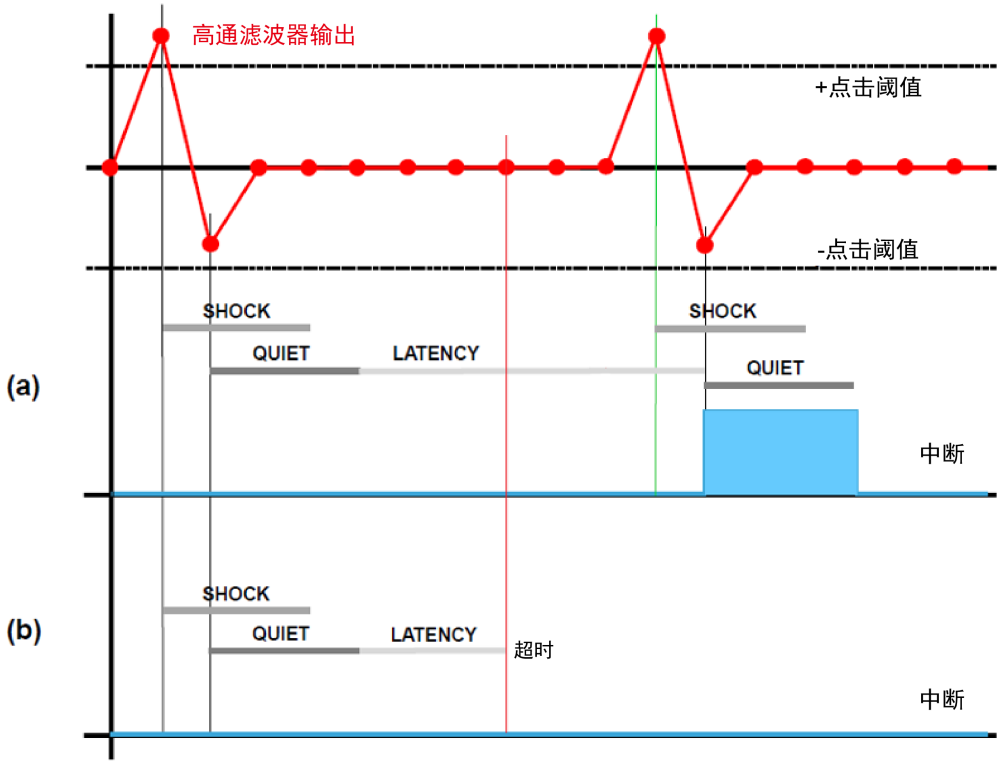
程。图 12. 双击事件识别（LIR 位 = 0）的情况（a）中，正确识别出了双击事件，而在情况（b）中，由于在经过 Latency 窗口间隔之后高通滤波数据超出了阈值，因此未产生中断。

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，高通滤波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 或 ALL_INT_SRC 寄存器被读取。

图 12. 双击事件识别（LIR 位 = 0）



5.6.3 单击和双击识别配置

可对 LIS2DW12 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：TAP_THS_Z 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。

TAP_THS_X、TAP_THS_Y、TAP_THS_Z 寄存器的 TAP_THSX[4:0]、TAP_THSY[4:0]、TAP_THSZ[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高通滤波数据上。

用户还可以通过寄存器 TAP_THS_Y 中的 TAP_PRIOR [2:0]位，定义单/双击中断事件的“优先级报告”。由于在源寄存器中仅给出与具有较高优先级的轴相关的点击事件，因此，当多个轴上同时出现单/双击事件时，这一功能非常实用。

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 LATENCY[3:0]位用来设置 latency 时间窗口值：这几个位的默认值为

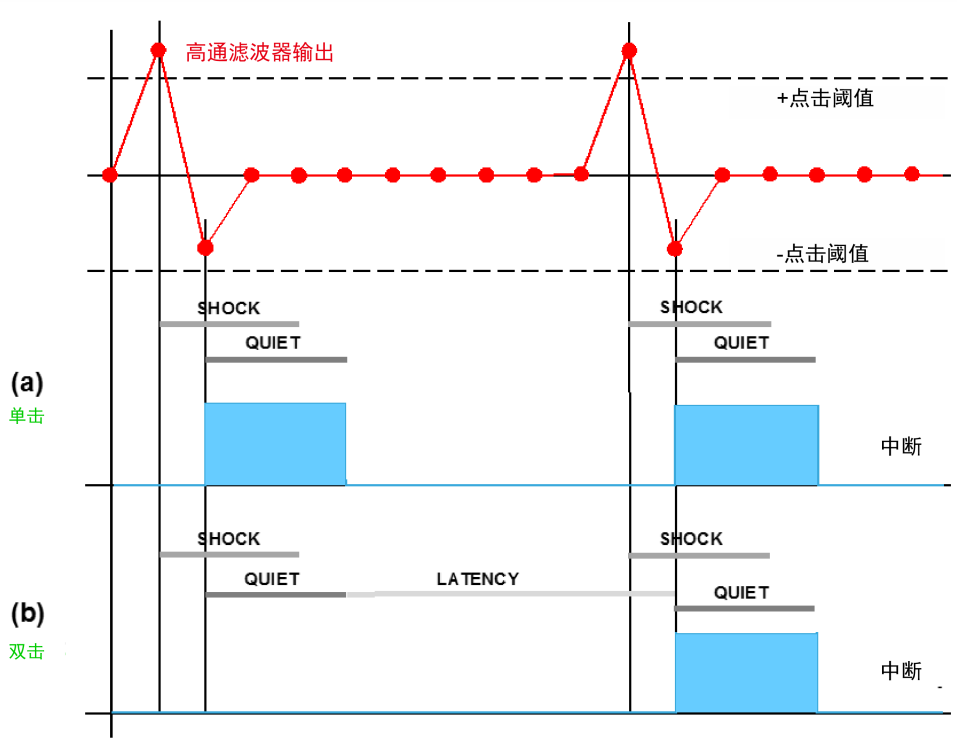
0000b，对应于 16*ODR_XL 的时间，这里 ODR_XL 为加速度计输出数据率。如果 LATENCY[3:0]位被置为其他不同的值，那么 1 LSB 对应于 32/ODR 的时间。

图 13. 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_INT1_PAD_CTRL 寄存器的 INT1_SINGLE_TAP 位置为 1 来实现，双击情况下通过将 CTRL4_INT1_PAD_CTRL 寄存器的 INT1_TAP 位置为 1 来实现。

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。

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 13. 单击和双击识别（LIR 位 = 0）



也可以通过读取 TAP_SRC 寄存器来检查点击中断信号，如表 20. TAP_SRC 寄存器所述。

表 20. 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 1。
- 当检测到单击时，SINGLE_TAP 置为 1。
- 当检测到双击时，DOUBLE_TAP 置为 1。
- 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 0，符号为负时它为 1。
- 当在 X（Y、Z）轴上检测到点击事件时，X_TAP（Y_TAP、Z_TAP）置为 1

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 位置为 1 时，使能单击和双击识别功能，并始终首先识别单击事件，然后再识别双击事件。

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 或 ALL_INT_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软件流程：

1. 将 74h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz，高性能
2. 将 04h 写入 CTRL6	// 使能 FS ±2 *g* LOW_NOISE
3. 将 09h 写入 TAP_THS_X	// 设置 X 轴的点击阈值
4. 将 E9h 写入 TAP_THS_Y	// 设置 Y 轴的点击阈值 // 设置 TAP 优先级 Z-Y-X
5. 将 E9h 写入 TAP_THS_Z	// 使能 X、Y、Z 轴上的点击检测 // 设置 Z 轴的点击阈值
6. 将 06h 写入 INT_DUR	// 设置 Quiet 和 Shock 时间窗口
7. 将 00h 写入 WAKE_UP_THS	// 只使能单击（SINGLE_DOUBLE_TAP = 0）
8. 将 40h 写入 CTRL4_INT1_PAD_CTRL	// 单击中断驱动至 INT1 引脚
9. 将 20h 写入 CTRL7	// 使能中断

本例中，各个轴的阈值被置为 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：当高通滤波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 74h 写入 CTRL1	// 启动加速度计 // ODR = 400 Hz，高性能
2. 将 04h 写入 CTRL6	// 使能 FS ±2 *g* LOW_NOISE
将 0Ch 写入 TAP_THS_X	// 设置 X 轴的点击阈值
将 ECh 写入 TAP_THS_Y	// 设置 Y 轴的点击阈值 // 设置 TAP 优先级 Z-Y-X
3. 将 ECh 写入 TAP_THS_Z	// 使能 X、Y、Z 轴上的点击检测 // 设置 Z 轴的点击阈值
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	// 使能单击和双击 // (SINGLE_DOUBLE_TAP = 1)
6. 将 08h 写入 TRL4_INT1_PAD_CTRL	// 单击中断驱动至 INT1 引脚
7. 将 20h 写入 CTRL7	// 使能中断

本例中，各个轴的阈值被置为 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，高通滤波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 时间窗口内高通滤波数据不能超阈值。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。

将 WAKE_UP_THS（34h）寄存器的 SLEEP_ON 位置为 1，使能活动/不活动功能。如果检测到睡眠状态条件，则 LIS2DW12 在 CTRL1（20h）中 LP_MODE [1:0]位先前选择的低功耗模式下自动进入 12.5 Hz ODR。一旦检测到唤醒事件，LIS2DW12 会从睡眠状态唤醒，切换到工作模式和 CTRL1（20h）寄存器配置的 ODR。

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全性能模式，反之亦然，因此可以保证节能和灵活性。

CTRL7 的 USR_OFF_ON_WU 位可以帮助活动/不活动识别功能选择使用高通滤波器或偏移输出，如图 6. 嵌入功能所示。

如果选择“偏移输出”，则每个轴都可以具有不同的偏移值，并写入寄存器 X_OFS_USR、Y_OFS_USR、Z_OFS_USR。位权重由寄存器 CTRL7 中的 USR_OFF_W 位定义。

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高通滤波数据幅度和时序进行编程（图 14. 活动/不活动识别（利用 HP 滤波器））。

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高通滤波数据。

当一定数量的连续 X、Y、Z 高通滤波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。

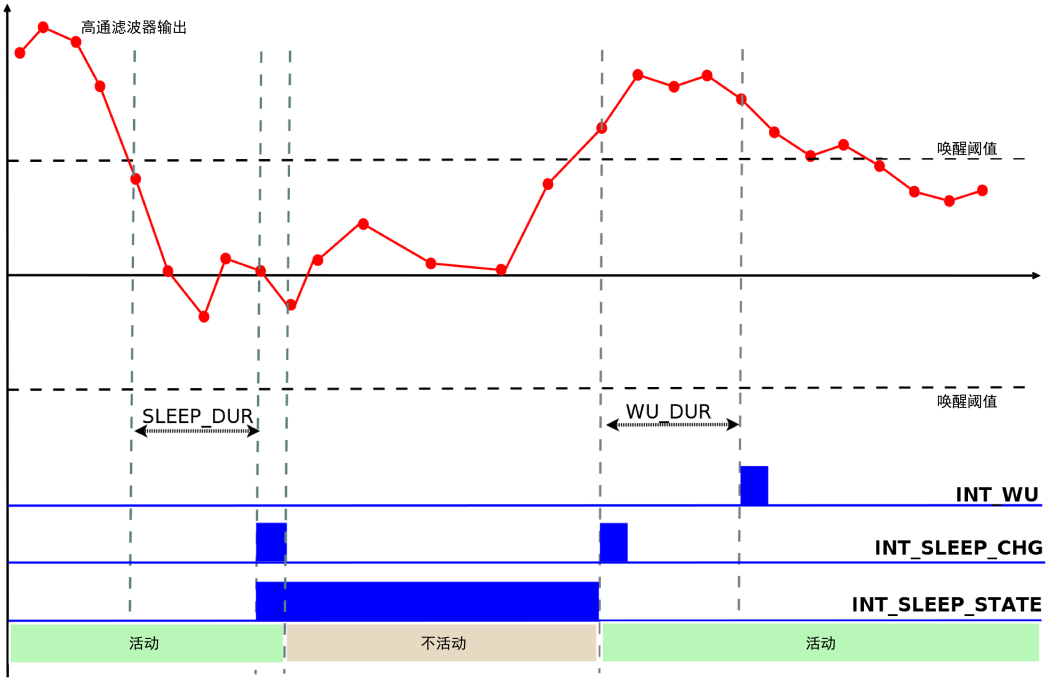
在设备处于不活动状态期间，STATUS 的 SLEEP_STATE 位设置为 1。该位可以使用 CTRL5_INT2_PAD_CTRL 中的 INT2_SLEEP_STATE 和 INT2_SLEEP_STATE_CHG 位，发送至 INT2 引脚，置为 1。

每当设备状态从活动变为不活动或反之时，ALL_INT_SRC 中的 SLEEP_CHANGE_IA 位设置为 1/ODR 时间。该位可以使用 CTRL5_INT2_PAD_CTRL 中的 INT2_SLEEP_CHG 位在 INT2 引脚上进行发送。可以使用 CTRL3 的 LIR 位，选择脉冲（1/ODR 脉冲持续时间）或锁存通知模式。

当一个轴上有一个采样的高通滤波数据大于阈值时，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_INT1_PAD_CTRL 寄存器的 INT1_WU 位设置为 1 来实现）。

图 14. 活动/不活动识别（利用 HP 滤波器）



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_INT1_PAD_CTRL
   // 活动（唤醒）中断驱动至 INT1 引脚
5. 将 20h 写入 CTRL7
   // 使能中断

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 高通滤波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）一个轴的高通滤波数据大于阈值，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）。

以下程序描述如何在 INT2 引脚上对睡眠变化事件进行发送：

1. 将 50h 写入 CTRL1
   // 启动加速度计
   // ODR = 200 Hz, FS = ±2 *g*
2. 将 02h 写入 WAKE_UP_DUR
   // 设置不活动检测的持续时间

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 40h 写入 CTRL5_INT2_PAD_CTRL // 睡眠变化中断驱动至 INT2 引脚
5. 将 20h 写入 CTRL7 // 使能中断

此示例与前一个示例类似，但发送的事件是 INT2 引脚上的“睡眠变化”。

5.8 静止/运动检测

静止/运动检测是“活动/不活动”功能的特殊情况，其中，当检测到睡眠条件（相当于静止条件）时，ODR/电源模式不改变。通过在 WAKE_UP_DUR 中将 STATIONARY 位置为 1 来激活静止/运动检测。如果 WAKE_UP_THS（34h）中的 STATIONARY 位和 SLEEP_ON 位均置为 1，则选择静止/运动检测。

5.9 启动状态

设备上电后，LIS2DW12 执行一段 20 ms 的启动程序来加载修整参数（寄存器地址：02h；从 07h 至 0Bh；从 10h 至 1Fh）。启动完成后，器件会自动配置为掉电模式。

启动时间内，寄存器不可访问。

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

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。如果需要复位到控制寄存器的默认值（寄存器地址：从 20h 至 25h；2Eh；从 30h 至 36h；从 3Ch 至 3Fh），可以将 CTRL2 寄存器的 SOFT_RESET 位置为 1。软件复位程序耗时 5 µs。

通过将 CTRL5_INT2_PAD_CTRL 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”。

必须按照以下示例中所示的顺序（从任意工作模式）执行流程：

1. 将 SOFT_RESET 位置为“1”
2. 等待 5 µs（或等待至 CTRL2 寄存器的 SOFT_RESET 位返回 0）；
3. 将 BOOT 位置为“1”
4. 等待 20 ms

为了避免冲突，重启和软件复位不能同时执行（不要同时将 CTRL2 寄存器的 BOOT 位和 SOFT_RESET 位同时置为 1）。

6 先进先出（FIFO）缓冲区

为了减少主机处理器干预并便于对事件识别数据进行后处理，LIS2DW12 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 完整事件在 INT1 或 INT2 引脚上生成专用中断。

6.1 FIFO 描述

FIFO 缓冲区能够根据寄存器 CTRL1 中的 MODE [1:0]位和 LP_MODE [1:0]位，以高分辨率存储多达 32 个加速度样本。

数据样本集合由 6 个字节（Xl、Xh、Yl、Yh、Zl 和 Zh）和组成，它们会以寄存器 CTRL1 的 ODR[3:0]所定义的选定输出数据速率释放到 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 21. FIFO 缓冲区填满示例（存储第 32 个采样集）

输出寄存器	28h	29h	2Ah	2Bh	2Ch	2Dh
	Xl	Xh	Yl	Yh	Zl	Zh
FIFO 索引	FIFO 样本集合					
FIFO（0）	Xl（0）	Xh（0）	Yl（0）	Yh（0）	Zl（0）	Zh（0）
FIFO（1）	Xl（1）	Xh（1）	Yl（1）	Yh（1）	Zl（1）	Zh（1）
FIFO（2）	Xl（2）	Xh（2）	Yl（2）	Yh（2）	Zl（2）	Zh（2）
FIFO（3）	Xl（3）	Xh（3）	Yl（3）	Yh（3）	Zl（3）	Zh（3）
...	...	...	...	...	...	...
FIFO（30）	Xl（30）	Xh（30）	Yl（30）	Yh（30）	Zl（30）	Zh（30）
FIFO（31）	Xl（31）	Xh（31）	Yl（31）	Yh（31）	Zl（31）	Zh（31）

表 22. FIFO 缓冲区完整表示（存储了第 33 个样本集合、丢弃了第 1 个样本）

输出寄存器	28h	29h	2Ah	2Bh	2Ch	2Dh
	Xl	Xh	Yl	Yh	Zl	Zh
FIFO 索引	样本集合					
FIFO（0）	Xl（1）	Xh（1）	Yl（1）	Yh（1）	Zl（1）	Zh（1）
FIFO（1）	Xl（2）	Xh（2）	Yl（2）	Yh（2）	Zl（2）	Zh（2）
FIFO（2）	Xl（3）	Xh（3）	Yl（3）	Yh（3）	Zl（3）	Zh（3）
FIFO（3）	Xl（4）	Xh（4）	Yl（4）	Yh（4）	Zl（4）	Zh（4）
...	...	...	...	...	...	...
FIFO（31）	Xl（32）	Xh（32）	Yl（32）	Yh（32）	Zl（32）	Zh（32）

表 21. FIFO 缓冲区填满示例（存储第 32 个采样集） 表 22. FIFO 缓冲区完整表示（存储了第 33 个样本集合、丢弃了第 1 个样本）表示的是存储了 32 个样本时 FIFO 已满的状态，而表示下一步，当第 33 个采样插入到 FIFO 中，同时第 1 个采样被覆盖。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，LIS2DW12 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 缓冲区由两个不同的加速度计寄存器进行管理，一个寄存器可使能并配置 FIFO 特性，另外一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 (2Eh)

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 FIFO 关闭；只要模式设置为 Bypass 以外的其他模式，FIFO 就会使能并开始存储采样。

表 23. FIFO_CTRL 寄存器

b7	b6	b5	b4	b3	b2	b1	b0
FMode2	FMode1	FMode0	FTH4	FTH3	FTH2	FTH1	FTH0

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

FTH[4:0]位请参见 Section 6.3.1 FIFO 阈值介绍。

6.2.2 FIFO_SAMPLES (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 24. FIFO_SAMPLES 寄存器

b7	b6	b5	b4	b3	b2	b1	b0
FIFO_ FTH	FIFO_ OVR	Diff5	Diff4	Diff3	Diff2	Diff1	Diff0

- 当 FIFO 内容大于或等于水印等级时，FIFO_FTH 位设置为 1。此标志可发送至 INT1 或 INT2 引脚（参见 Section 6.3 FIFO 中断）。
- 在 FIFO 缓冲区满后，重写第一个样本时，FIFO_OVR 位置为 1。这意味着 FIFO 缓冲区包含 32 个未读样本。第一个样本集合已被读取时，FIFO_OVR 位会复位。
- Diff5 位与 Diff[4:0]位一起使用，可提供用了多少 FIFO 空间的信息（000000b 表示 FIFO 为空，100000b 表示 FIFO 已满）。此标志可发送至 INT1 或 INT2 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 25. FIFO_SAMPLES 特性（假定 FTH[4:0] = 15）

FIFO_FTH	Diff5（FIFO_FULL）	FIFO_OVR	Diff[4:0]	未读 FIFO 样本	时序
0	0	0	00000	0	t0
0	0	0	00001	1	t0 + 1/ODR
0	0	0	00010	2	t0 + 2/ODR
...	...	...	...	...	...
0	0	0	01110	14	t0 + 14/ODR
1	0	0	01111	15	t0 + 15/ODR
...	...	...	...	...	...
1	0	0	11111	31	t0 + 31/ODR
1	1	0	00000	32	t0 + 32/ODR
1	1	1	00000	32	t0 + 33/ODR

6.3 FIFO 中断

有三个特定的 FIFO 事件可以传输到引脚，以中断主处理器：FIFO 阈值、FIFO 已满和 FIFO 溢出。

所有 FIFO 事件都可以发送至 INT1 和 INT2 引脚。

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_CTRL 寄存器中的 FTH [4:0]字段在 0 到 31 之间的范围内选择所需的等级。

如果 FIFO（Diff[5:0]）中的条目数大于或等于 FTH [4:0]中编程的值，则 FIFO_SAMPLES 寄存器中的 FIFO_FTH 位置为高电平。

Diff[5:0]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_INT1_PAD_CTRL 寄存器的 INT1_FTH 位和 CTRL5_INT2_PAD_CTRL 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 CTRL4_INT1_PAD_CTRL 寄存器的 INT1_DIFF5 位置为 1（或将 CTRL5_INT2_PAD_CTRL 寄存器的 INT2_DIFF5 位置为 1）。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。

6.3.3 FIFO 溢出

如果 FIFO 发生溢出事件，则可以将设备配置为产生中断。为此，只需将 CTRL5_INT2_PAD_CTRL 寄存器的 INT2_OVR 位置为 1 即可。

6.4 FIFO 模式

通过 FIFO_CTRL 寄存器的 FMODE[2:0]字段，LIS2DW12 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。

6.4.1 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。

6.4.2 FIFO 模式

FIFO 模式中，缓冲区继续填充直至充满（存储了 32 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[5:0]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 32 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。

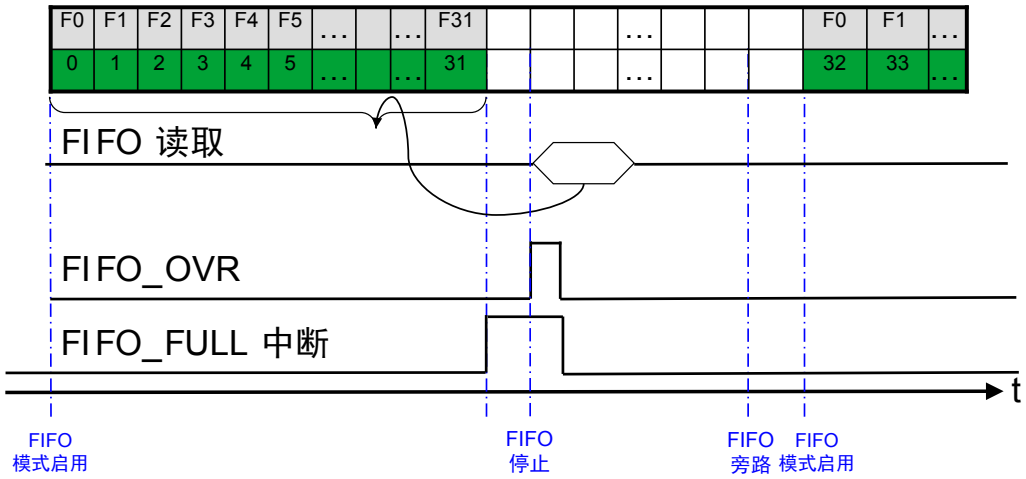
要尽快处理 FIFO 完整（Diff5 位）事件，建议将其传送到引脚，以产生一个中断，该中断随后会由一个特定的处理程序进行管理：

1. 将 INT1_DIFF5 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 15. FIFO 模式特性



如图 15. 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 32 个位置（从 F0 到 F31）。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（Diff5）触发的中断处理程序内执行，它包括 32 个 6 字节样本集合（共 192 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。

6.4.3 连续模式

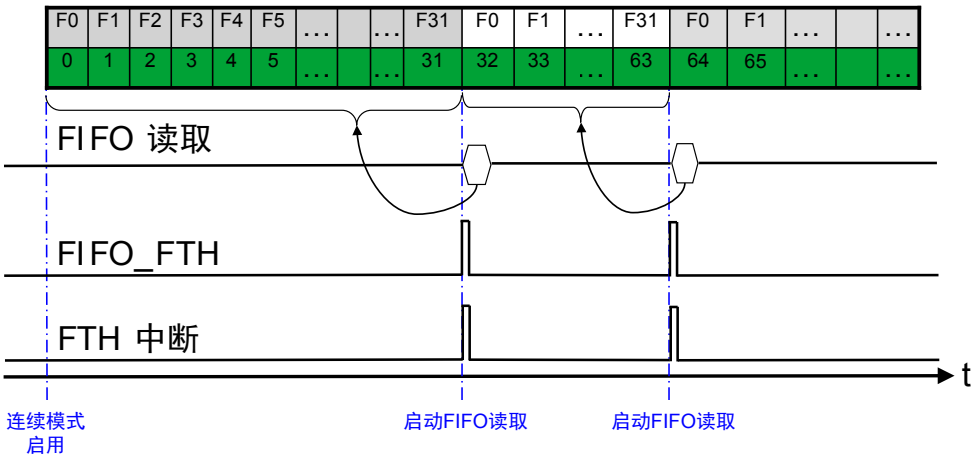
在 Continuous 模式下，FIFO 会继续填入数据，如果缓冲区已满，FIFO 索引会从头开始使用，较早的数据会被当前数据替代。最早先的数据继续被覆盖，直至读取操作释放了 FIFO 插槽。为了使 FIFO 位置的释放速度快于获得新数据的速度，主机处理器的读取速度至关重要。Bypass 配置下 FMODE[2:0]用于停止该模式。

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[4:0]置为 31。
2. 将 INT1_FTH 置为‘1’：使能 FIFO 阈值中断
3. 将 FIFO_CTRL 寄存器（2Eh）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 16. 带中断触发的 Continuous 模式



如图 16. 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 F31）。当缓冲区满时，FTH 中断（以及 FIFO_SAMPLES（2Fh）中的 Diff5 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（32 * 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 6.5 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。

6.4.4 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。

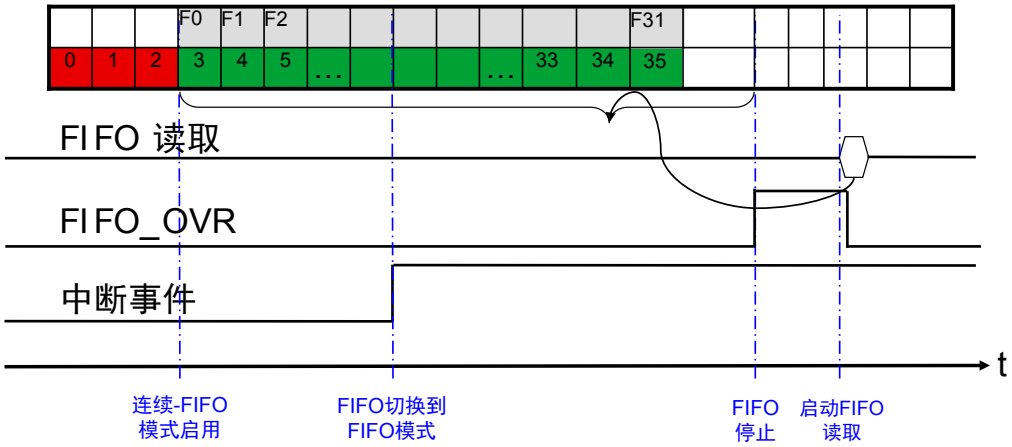
请按照以下步骤配置 Continuous-FIFO 模式：

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（2Eh）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 17. Continuous-FIFO 模式：中断锁存和非锁存）。

图 17. Continuous-FIFO 模式：中断锁存和非锁存



6.4.5 旁路-连续模式

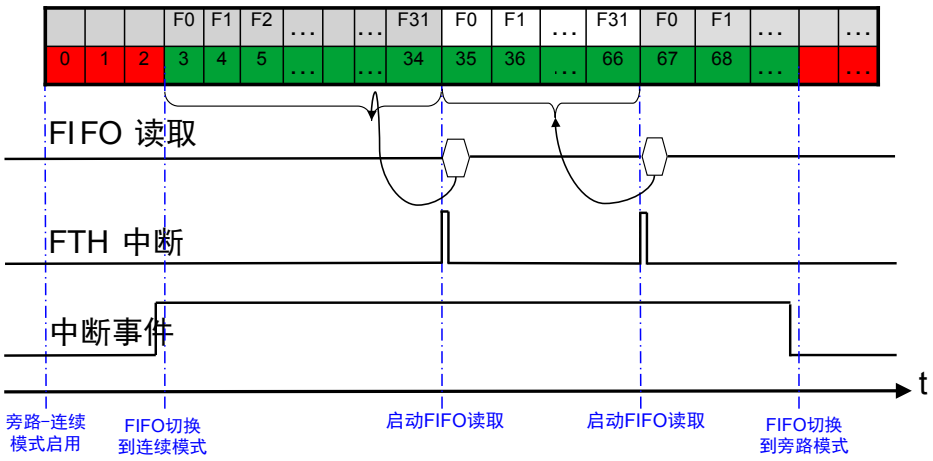
此模式是先前所述的旁路和连续模式的组合。在 Bypass-Continuous 模式下，FIFO 缓冲区会开始处于 Bypass 模式，在发生选定的中断（例如，唤醒、自由落体、点击……）后切换为 Continuous 模式。

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[4:0]置为 31。
3. 将 INT1_FTH 置为‘1’：使能 FIFO 阈值中断
4. 将 FIFO_CTRL 寄存器（2Eh）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 18. 旁路-连续模式


F0
F1
F2
F31
F0
F1
F31
F0
F1
0
1
2
3
4
5
34
35
36
66
67
68
FIFO 读取
FTH 中断
中断事件
t
旁路-连续模式启用
FIFO切换到连续模式
启动FIFO读取
启动FIFO读取
FIFO切换到旁路模式

如图 18. 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（32 * 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。

6.5 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样本，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 32 次读取操作，可以重新取回整个 FIFO 内容。存储在 FIFO 中的数据大小取决于所选的功耗模式。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：192 次单字节读取，32 次 6 字节读取，1 次 192 字节的多字节读取等）。

建议以 192 字节的多字节读取（6 个输出寄存器乘以 32 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 17.57 ms（29 + 9 * 192 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 772.5 µs。

如果按照此建议，使用标准 I²C（100 kHz），那么全部读取 FIFO（17.57 ms）将需要耗费 28/ODR，ODR 为 1600 Hz。使用 SPI @ 2 MHz（设备支持的最大速率是 10 MHz）时，全部读取 FIFO 的耗时将是数据生成时间的 2 倍（2*1/ODR），其中，ODR 为 1600 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 26. 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	32	32	32
100	17	32	32
200	8	32	32
400	4	17	32
800	1	8	32
1600	-	4	25

7 温度传感器

LIS2DW12 具有内部温度传感器，适用于环境温度测量。

如果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。

当一组新数据可用时，STATUS_DUP（37h）中的 DRDY_T 位设为 1，并在读取其中一个温度数据输出（OUT_T_H 或 OUT_T）时复位。DRDY_T 位可以通过 CTRL5_INT2_PAD_CTRL 寄存器的位 INT2_DRDY_T 在 INT2 引脚上进行发送。

可以使用 CTRL7 寄存器的 DRDY_PULSED 位进行温度 DRDY 中断的脉冲调制。

温度数据表示为二进制补码格式的 12 位，在 OUT_T_L 和 OUT_T_H 寄存器中左对齐。也可以使用寄存器 OUT_T 的 OUT_T_H 重复值，以便以二进制补码格式提供 8 位，其中，温度可按传感器输出的顺序进行读取。有关温度传感器的详细信息，请参见下表。

表 27. 温度传感器特性

符号	参数	最小值	典型(1)	最大值	单位
TsDr	温度传感器输出变化 vs.温度		1(2)		LSB/°C
			16(3)		
TODR	高性能模式或低功耗模式下的所有 ODR 的温度刷新率均等于 200/100/50 Hz		50		Hz
	低功耗模式下的 ODR 的温度刷新率等于 25 Hz		25		
	低功耗模式下的 ODR 的温度刷新率等于 12.5 Hz		12.5		
	低功耗模式下的 ODR 的温度刷新率等于 1.6 Hz		1.6		

1. *不保证典型规格参数。*
2. *8 位分辨率（即，在使用 OUT_T 寄存器时）*
3. *12 位分辨率（即，在使用 OUT_T_L 和 OUT_T_H 寄存器时）*

7.1 温度数据计算示例

表 28. 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 28. 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。

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。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，传感器输出电平由静电测试力和重力产生的数据的代数和给出。

该过程包括：

1. 使能加速度计
2. 启动自检之前，对五个样本取平均
3. 启动自检之后，对五个样本取平均
4. 计算每个轴的绝对值，并验证它是否处于给定的范围内。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 19. 加速度计自检步骤中所示。

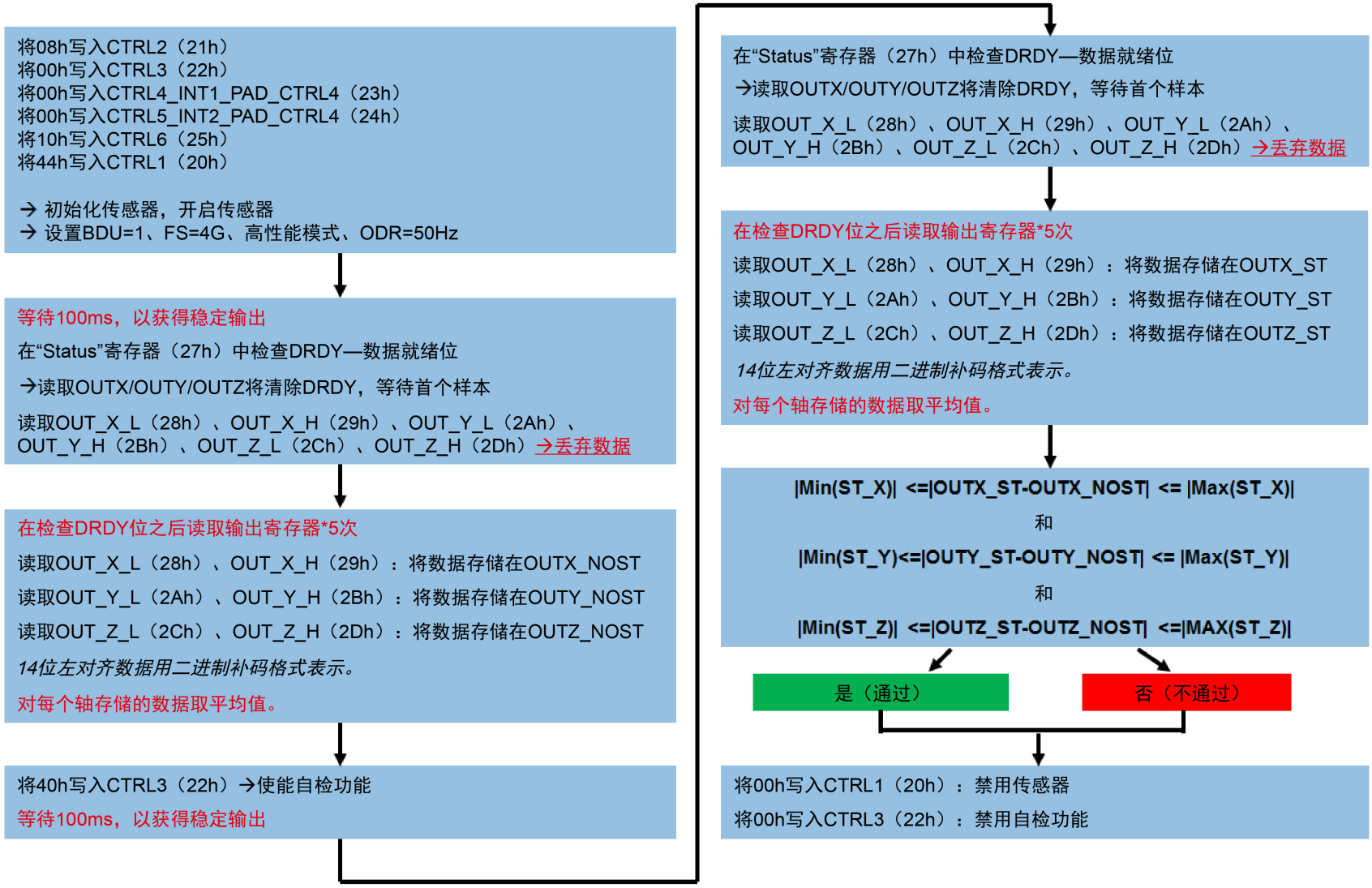
注意：

1. 自检过程中保持器件静止。

2. 自检程序中使用的满量程和数据速率并非强制性值，但建议使用。

3. 有关最小值和最大值，请参阅数据表。

图 19. 加速度计自检步骤



请注意：100 ms 的等待时间并非强制性，但建议使用。无论如何，应考虑建立时间。

版本历史

表 29. 文档版本历史

日期	版本	变更
2017 年 9 月 20 日	1	初始版本
2018 年 6 月 01	2	在 增加了有关 SDO/SA0 引脚的脚注 表 1. 内部引脚状态 更新了 表 8. ODR = 200 Hz 时的噪音[µg/√Hz] 更新了 Section 3.4 加速度计带宽 更新了 Section 4.1 启动序列 更新了 Section 4.5.1 输出数据示例 更新了 图 6. 嵌入功能 更新了 表 16. 自由落体阈值 更新了 Section 5.4 唤醒中断 更新了 Section 5.7 活动/不活动识别 更新了 Section 5.9 启动状态 更新了 Section 6.2.2 FIFO_SAMPLES (2Fh) 更新了 图 15. FIFO 模式特性 更新了 图 16. 带中断触发的 Continuous 模式 更新了 图 17. Continuous-FIFO 模式：中断锁存和非锁存 更新了 Section 8 自检功能

目录

表一览

图一览