# LIS2DH12: MEMS 数字输出运动传感器超低功耗高性能 3 轴“nano”加速度计

## 引言

本文档介绍了以 LGA 封装提供的低压 3 轴数字量输出线性 MEMS 加速度计。

LIS2DH12 是属于“nano”系列的超低功耗高性能 3 轴线性加速度计，具有数字 $I^2C$、SPI 串行接口标准输出。

器件具有超低功耗工作模式，可实现高级节能、智能睡眠唤醒以及恢复睡眠功能。

LIS2DH12 具有±2*g*/±4*g*/±8*g*/±16*g* 的动态用户可选满量程，并能通过 1 Hz 到 5 kHz 的输出数据速率测量加速度。

器件可配置为通过独立的惯性唤醒/自由落体事件以及通过器件自身的位置生成中断信号。中断发生器的阈值和时序可由终端用户动态设定。

也可通过可自动编程的睡眠唤醒和恢复睡眠功能提高节能效率。

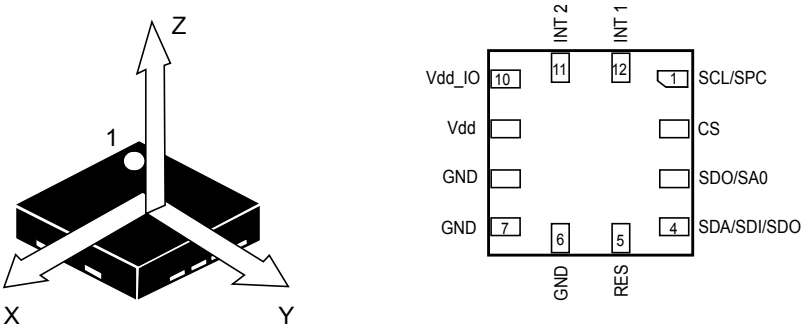
LIS2DH12 集成了 32 级的先进先出（FIFO）缓冲器，允许用户进行数据存储，可限制主机处理器的干预。

LIS2DH12 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




（顶视图）
可检测加速度的方向

（底视图）

表 1. 内部引脚状态

| 引脚# | 名称 | 功能 | 引脚状态 |
|---|---|---|---|
| 1 | SCL<br>SPC | I²C 串行时钟（SCL）<br>SPI 串口时钟（serial port clock，SPC） | 默认：输入高阻抗 |
| 2 | CS | SPI 使能 (SPI enable)<br>I²C/SPI 模式选择：<br>1：SPI 空闲模式/ I²C 通信使能<br>0：SPI 通信模式/ I²C 禁用 | 默认：输入高阻抗 |
| 3 | SDO<br>SA0 | SPI 串行数据输出（SDO）<br>器件地址的 I²C 较低有效位（SA0） | 默认值：带内部上拉的输入(1) |
| 4 | SDA<br>SDI<br>SDO | I²C 串行数据（SDA）<br>SPI 串行数据输入（serial data input，SDI）<br>3 线接口串行数据输出（serial data output，SDO） | 默认值：（SDA）输入高阻抗 |
| 5 | Res | 与 GND 连接 | |
| 6 | GND | 0 V 电源 | |
| 7 | GND | 0 V 电源 | |
| 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 | 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. *为了禁用 SDO/SA0 引脚上的内部上拉，在 CTRL_REG0（1Eh）中写入 90h。*

# 2 寄存器

表 2. 寄存器

| 寄存器名 | 地址 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| STATUS_REG_AUX | 07h | - | TOR | - | - | - | TDA | - | - |
| RESERVED | 08h -0Bh | | | | | | | | |
| OUT_TEMP_L | 0Ch | Temp7 | Temp6 | Temp5 | Temp4 | Temp3 | Temp2 | Temp1 | Temp0 |
| OUT_TEMP_H | 0Dh | Temp15 | Temp14 | Temp13 | Temp12 | Temp11 | Temp10 | Temp9 | Temp8 |
| RESERVED | 0Eh | | | | | | | | |
| WHO_AM_I | 0Fh | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| CTRL_REG0 | 1Eh | SDO_PU_DISC | 0 (1) | 0(1) | 1(2) | 0(1) | 0(1) | 0(1) | 0(1) |
| TEMP_CFG_REG | 1Fh | TEMP_EN1 | TEMP_EN2 | 0 | 0 | 0 | 0 | 0 | 0 |
| CTRL_REG1 | 20h | ODR3 | ODR2 | ODR1 | ODR0 | LPen | Zen | Yen | Xen |
| CTRL_REG2 | 21h | HPM1 | HPM0 | HPCF2 | HPCF1 | FDS | HPCLICK | HP_IA2 | HP_IA1 |
| CTRL_REG3 | 22h | I1_CLICK | I1_IA1 | I1_IA2 | I1_ZYXDA | 0(1) | I1_WTM | I1_溢出 | - |
| CTRL_REG4 | 23h | BDU | BLE | FS1 | FS0 | HR | ST1 | ST0 | SIM |
| CTRL_REG5 | 24h | BOOT | FIFO_EN | - | - | LIR_INT1 | D4D_INT1 | LIR_INT2 | D4D_INT2 |
| CTRL_REG6 | 25h | I2_CLICK | I2_IA1 | I2_IA2 | I2_BOOT | I2_ACT | - | INT_POLARITY | - |
| REFERENCE | 26h | REF7 | REF6 | REF5 | REF4 | REF3 | REF2 | REF1 | REF0 |
| STATUS_REG | 27h | ZYXOR | ZOR | YOR | XOR | ZYXDA | ZDA | YDA | XDA |
| OUT_X_L | 28h | XD7 | XD6 | XD5 | XD4 | XD3 | XD2 | XD1 | XD0 |
| OUT_X_H | 29h | XD15 | XD14 | XD13 | XD12 | XD11 | XD10 | XD9 | XD8 |
| OUT_Y_L | 2Ah | YD7 | YD6 | YD5 | YD4 | YD3 | YD2 | YD1 | YD0 |
| OUT_Y_H | 2Bh | YD15 | YD14 | YD13 | YD12 | YD11 | YD10 | YD9 | YD8 |
| OUT_Z_L | 2Ch | ZD7 | ZD6 | ZD5 | ZD4 | ZD3 | ZD2 | ZD1 | ZD0 |
| OUT_Z_H | 2Dh | ZD15 | ZD14 | ZD13 | ZD12 | ZD11 | ZD10 | ZD9 | ZD8 |
| FIFO_CTRL_REG | 2Eh | FM1 | FM0 | TR | FTH4 | FTH3 | FTH2 | FTH1 | FTH0 |
| FIFO_SRC_REG | 2Fh | WTM | OVRN_FIFO | 空 | FSS4 | FSS3 | FSS2 | FSS1 | FSS0 |
| INT1_CFG | 30h | AOI | 6D | ZHIE | ZLIE | YHIE | YLIE | XHIE | XLIE |
| INT1_SRC | 31h | 0 | IA | ZH | ZL | YH | YL | XH | XL |
| INT1_THS | 32h | 0 | THS6 | THS5 | THS4 | THS3 | THS2 | THS1 | THS0 |
| INT1_DURATION | 33h | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| INT2_CFG | 34h | AOI | 6D | ZHIE | ZLIE | YHIE | YLIE | XHIE | XLIE |
| INT2_SRC | 35h | 0 | IA | ZH | ZL | YH | YL | XH | XL |
| INT2_THS | 36h | 0 | THS6 | THS5 | THS4 | THS3 | THS2 | THS1 | THS0 |
| INT2_DURATION | 37h | 0 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CLICK_CFG | 38h | - | - | ZD | ZS | YD | YS | XD | XS |
| CLICK_SRC | 39h | - | IA | DCLICK | SCLICK | 符号 | Z | 是 | X |

| 寄存器名 | 地址 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| CLICK_THS | 3Ah | LIR_CLICK | Ths6 | Ths5 | Ths4 | Ths3 | Ths2 | Ths1 | Ths0 |
| TIME_LIMIT | 3Bh | - | TLI6 | TLI5 | TLI4 | TLI3 | TLI2 | TLI1 | TLI0 |
| TIME_LATENCY | 3Ch | TLA7 | TLA6 | TLA5 | TLA4 | TLA3 | TLA2 | TLA1 | TLA0 |
| TIME_WINDOW | 3Dh | TW7 | TW6 | TW5 | TW4 | TW3 | TW2 | TW1 | TW0 |
| ACT_THS | 3Eh | - | Acth6 | Acth5 | Acth4 | Acth3 | Acth2 | Acth1 | Acth0 |
| INACT_DUR | 3Fh | ActD7 | ActD6 | ActD5 | ActD4 | ActD3 | ActD2 | ActD1 | ActD0 |

1. *为了设备的正确运行，此位必须置为0。*
2. *为了设备的正确运行，此位必须置为1。*

# 3 工作模式

LIS2DH12 提供四种不同的工作模式：掉电模式、高分辨率/正常模式和低功耗模式。正常模式可确保达到更高的分辨率，而低功耗模式可以进一步减少电流消耗。

施加电源后，LIS2DH12 执行一段 5 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 LIS2DH12 数据表，CTRL_REG1 的输出数据率（ODR）和低功耗使能（LPen）位以及 CTRL_REG4 的 HR 位用于选择工作模式（掉电模式、高分辨率/正常模式和低功耗模式）以及输出数据率（表 3. 工作模式选择和表 4. 数据速率配置）。

**表 3. 工作模式选择**

| 工作模式 | CTRL_REG1[3]（LPen 位） | CTRL_REG4[3]（HR 位） | BW [Hz] | 导通时间[ms] | 灵敏度 @ ±2*g* [mg/位] |
|---|---|---|---|---|---|
| 低功耗模式（8 位数据输出） | 1 | 0 | ODR/2 | 1 | 16 |
| 正常模式（10 位数据输出） | 0 | 0 | ODR/2 | 1.6 | 4 |
| 高分辨率模式（12 位数据输出） | 0 | 1 | ODR/9 | 7/ODR | 1 |
| 不允许 | 1 | 1 | -- | -- | -- |

**表 4. 数据速率配置**

| ODR3 | ODR2 | ODR1 | ODR0 | 功率模式选择 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 掉电模式 |
| 0 | 0 | 0 | 1 | 高分辨率/正常/低功耗模式（1 Hz） |
| 0 | 0 | 1 | 0 | 高分辨率/正常/低功耗模式（10 Hz） |
| 0 | 0 | 1 | 1 | 高分辨率/正常/低功耗模式（25 Hz） |
| 0 | 1 | 0 | 0 | 高分辨率/正常/低功耗模式（50 Hz） |
| 0 | 1 | 0 | 1 | 高分辨率/正常/低功耗模式（100 Hz） |
| 0 | 1 | 1 | 0 | 高分辨率/正常/低功耗模式（200 Hz） |
| 0 | 1 | 1 | 1 | 高分辨率/正常/低功耗模式（400 Hz） |
| 1 | 0 | 0 | 0 | 低功耗模式(1.60 kHz) |
| 1 | 0 | 0 | 1 | 正常(1.344 kHz)/低功耗模式(5.376 kHz) |

表 5. 工作模式的电流消耗 列出了不同工作模式下的典型功耗值。

**表 5. 工作模式的电流消耗**

| 工作模式 [Hz] | 低功耗模式（8 位数据输出） [µA] | 正常模式（10 位数据输出） [µA] | 高分辨率（12 位数据输出） [µA] |
|---|---|---|---|
| 1 | 2 | 2 | 2 |
| 10 | 3 | 4 | 4 |

| 工作模式 [Hz] | 低功耗模式<br>（8 位数据输出）<br>[µA] | 正常模式<br>（10 位数据输出）<br>[µA] | 高分辨率<br>（12 位数据输出）<br>[µA] |
|---|---|---|---|
| 25 | 4 | 6 | 6 |
| 50 | 6 | 11 | 11 |
| 100 | 10 | 20 | 20 |
| 200 | 18 | 38 | 38 |
| 400 | 36 | 73 | 73 |
| 1344 | -- | 185 | 185 |
| 1620 | 100 | -- | -- |
| 5376 | 185 | -- | -- |

## 3.1 掉电模式

器件处于掉电模式时，器件内部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，因此可保持进入省电模式前存储器中采样的最后数据。

## 3.2 低功耗模式

在低功耗模式下，会以通过 ODR 位选择的数据速率生成数据，数据会用于通过 CTRL_REG1 的 Zen、Yen 和 Xen 位使能的轴。为已禁用的轴生成的数据为 00h。

在低功耗模式下，加速度数据分辨率为 8 位，左对齐，并保存在 OUT 寄存器中（从 28h 至 2D）。

为了使能低功耗模式，将 CTRL_REG4 中的 HR 位清零并将 CTRL_REG1 中的 LPen 位置位。

数据中断生成激活并通过 INT1_CFG 和 INT2_CFG 配置，并可以通过 CTRL_REG3 和 CTRL_REG6 寄存器连接到 INT1 或 INT2 引脚。

## 3.3 正常模式

在正常模式下，会以通过 ODR 位选择的数据速率生成数据，数据会用于通过 CTRL_REG1 的 Zen、Yen 和 Xen 位使能的轴。为已禁用的轴生成的数据为 00h。

在正常模式下，加速度数据分辨率为 10 位，左对齐，并保存在 OUT 寄存器中（从 28h 至 2D）。

为了使能正常模式，将 CTRL_REG4 中的 HR 位和 CTRL_REG1 中的 LPen 位清零。

数据中断生成激活并通过 INT1_CFG 和 INT2_CFG 配置，并可以通过 CTRL_REG3 和 CTRL_REG6 寄存器连接到 INT1 或 INT2 引脚。

## 3.4 高分辨率模式

在高分辨率模式下，会以通过 ODR 位选择的数据速率生成数据，数据会用于通过 CTRL_REG1 的 Zen、Yen 和 Xen 位使能的轴。为已禁用的轴生成的数据为 00h。

在高分辨率模式下，加速度数据分辨率为 12 位，左对齐，并保存在 OUT 寄存器中（从 28h 至 2D）。

为了使能高分辨率模式，将 CTRL_REG4 中的 HR 位置位并将 CTRL_REG1 中的 LPen 位清零。

数据中断生成激活并通过 INT1_CFG 和 INT2_CFG 配置，并可以通过 CTRL_REG3 和 CTRL_REG6 寄存器连接到 INT1 或 INT2 引脚。

建议在设备操作模式从高分辨率配置（HR）切换至掉电模式（PD）时读取寄存器 REFERENCE（26h）；此操作将在再次切换至正常/高性能模式前复位滤波模块：

1. 将 08h 写入 CTRL_REG4 // HR 置位
   // LPen 清零
2. 将 57h 写入 CTRL_REG2 // 使能所有轴
   // ODR = 100 Hz

3. 等待导通时间结束
4. 将 07h 写入 CTRL_REG2
   // LPen 清零
   // 使能所有轴
   // 掉电
5. 读取 REFERENCE
   // 复位滤波器模块
   // LPen 清零
6. 将 57h 写入 CTRL_REG2
   // 使能所有轴
   // ODR = 100 Hz
7. 等待导通时间结束

## 3.5 切换模式

表 6. 操作模式转换的导通时间中给出了转换到另一种操作模式的导通时间。

**表 6. 操作模式转换的导通时间**

| 工作模式改变 | 生效时间 [ms] |
|---|---|
| 12 位模式至 8 位模式 | 1/ODR |
| 12 位模式至 10 位模式 | 1/ODR |
| 10 位模式至 8 位模式 | 1/ODR |
| 10 位模式至 12 位模式 | 7/ODR |
| 8 位模式至 10 位模式 | 1/ODR |
| 8 位模式至 12 位模式 | 7/ODR |

# 4 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 5 毫秒后，器件会自动进入掉电模式。要导通器件并采集加速度数据，选择 CTRL_REG4 中的 HR 位和 CTRL_REG1 中的 LPen 位，使能至少一个轴并选择首选 ODR。

可使用下列通用序列对器件进行配置：

1. 写 CTRL_REG1
2. 写 CTRL_REG2
3. 写 CTRL_REG3
4. 写 CTRL_REG4
5. 写 CTRL_REG5
6. 写 CTRL_REG6
7. 写 REFERENCE
8. 写 INTx_THS
9. 写 INTx_DUR
10. 写 INTx_CFG
11. 写 CTRL_REG5

## 4.1 读取加速度数据

### 4.1.1 使用状态寄存器

LIS2DH12 连续生成数据（先 X，其次 Y，最后 Z）。每次生成单轴数据时，STATUS_REG 的相应 DA 信号（XDA、YDA、ZDA）位置为 1。在读取数据各自的高字节时，XDA、YDA 和 ZDA 分别重置为 0（即，在读取数据（寄存器 29h）的 X 轴高字节时，XDA 立即变为 0，以此类推）。

如果 XDA、YDA 和 ZDA 位同时为“1”，则仅在 Z 数据生成并置位后，ZYXDA 信号才可以置位，否则当 XDA、YDA 和 ZDA 同时为“0”时，ZYXDA 位重置为“0”。

在生成相应数据且相应 DA 位已置为“1”时，上溢标志 XOR、YOR、ZOR 位分别置为“1”，而在读取相应数据时，它们会被重置为“0”（与相应 DA 位一起）。

当至少一个上溢标志（XOR、YOR、ZOR）变为 1 时，ZYXOR 位置为 1，而当所有上溢标志为 0 时，ZYXOR 位重置为 0。

**表 7. STATUS_REG**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ZYXOR | ZOR | YOR | XOR | ZYXDA | ZDA | YDA | XDA |

**表 8. STATUS_REG 说明**

| | |
|---|---|
| ZYXOR | X、Y 和 Z 轴数据上溢。默认值：0<br>（0：未发生上溢；<br>1：一组新数据覆盖了上一组数据） |
| ZOR | Z 轴数据上溢。默认值：0<br>（0：未发生上溢；<br>1：Z 轴的一个新数据覆盖了上一个数据） |
| YOR | Y 轴数据上溢。默认值：0<br>（0：未发生上溢；<br>1：Y 轴的一个新数据覆盖了上一个数据） |

| | |
|---|---|
| XOR | X 轴数据上溢。默认值：0<br>（0：未发生上溢；<br>1：X 轴的一个新数据覆盖了上一个数据） |
| ZYXDA | X、Y 和 Z 轴有新数据可用。默认值：0<br>（0：一组新数据尚不可用；1：有一组新数据可用） |
| ZDA | Z 轴有新数据可用。默认值：0<br>（0：Z 轴的新数据尚不可用；<br>1：Z 轴有新数据可用） |
| YDA | Y 轴有新数据可用。默认值：0<br>（0：Y 轴的新数据尚不可用；<br>1：Y 轴有新数据可用） |

应对为器件提供的 STATUS_REG 进行轮询，以检查何时有可用的新数据集。读取数据的步骤如下：

1. 读取 STATUS_REG
2. 如果 STATUS_REG(3) = 0，则转至 1
3. 如果 STATUS_REG(7) = 1，说明一些数据已被重写
4. 读 OUTX_L
5. 读 OUTX_H
6. 读 OUTY_L
7. 读 OUTY_H
8. 读 OUTZ_L
9. 读 OUTZ_H
10. 数据处理
11. 调到步骤 1

第 3 步中进行的检查可了解读取速率是否适合数据生产速率。如果一个或多个加速度样本已被新数据覆盖，由于读取速率不够快，STATUS_REG 的 ZYXOR 位会置 1。

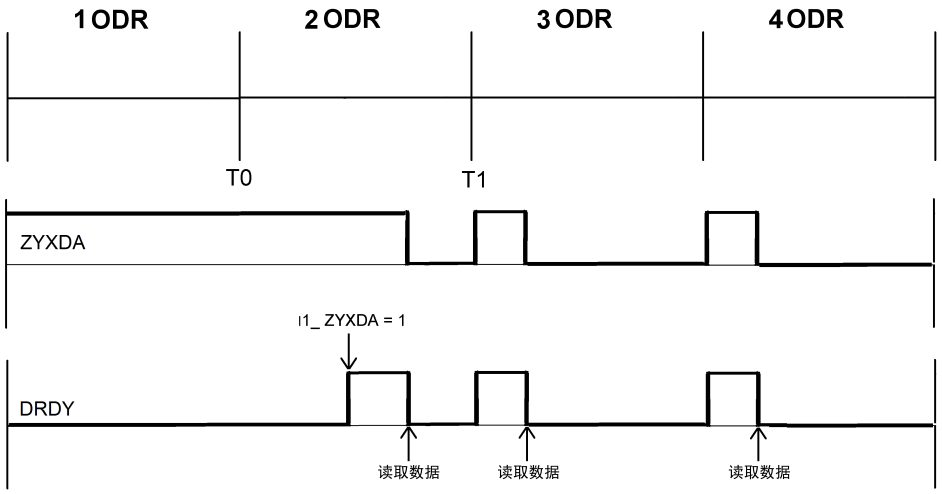
如果器件内存在的所有数据均已被读取，同时尚未生成新数据，上溢位会自动清零。

### 4.1.2 使用数据就绪（DRDY）信号

当 CTRL_REG3（22h）的 I1_ ZYXDA 位置为“1”时，可将 ZYXDA 驱动至 INT1 引脚。

此信号称为 DRDY。DRDY 的信号特性与 ZYXDA 位（图 2. DRDY 信号同步）的相同。

图 2. DRDY 信号同步

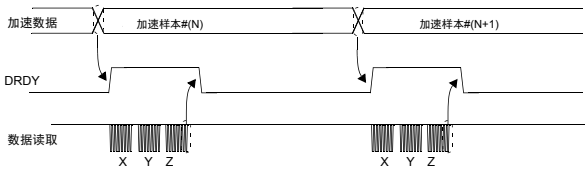



为确保第一个 DRDY 上升沿与所选 ODR（避免图 2. DRDY 信号同步中的情况）同步，在使能 ODR 前将 I1_ZYXDA 位置为“1”。

可通过 CTRL_REG6 的 INT_POLARITY 将 DRDY 信号的极性设置更改为低电平有效或高电平有效。当一组新的加速度数据生成并可读取时，数据准备就绪信号升高为“1”。当所有使能通道的数据的高字节已被读取时（29h、2Bh、2Dh），DRDY 复位。

*注意：CTRL_REG5 的 LIR_INT1 位不作用于 DRDY 信号。*

图 3. 数据准备就绪信号




### 4.1.3 使用块数据更新（block data update，BDU）功能

如果加速度数据的读取速度特别慢并且不能通过 STATUS_REG 中存在的 XYZDA 位或通过 DRDY 信号进行同步（或者不需要同步），则强烈建议将 CTRL_REG4 中的 BDU（块数据更新）位置 1。

此功能可避免读取与其它样本相关联的数值（加速度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加速度数据，但如果发起了对给定寄存器对（即 OUT_X_H 和 OUT_X_L、OUT_Y_H 和 OUT-Y_L、OUT_Z_H 和 OUT_Z_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 仅会确保已同时对 OUT_X(Y, Z)_L 和 OUT_X(X,Z)_H 进行采样。例如，如果读取速度过慢，可能会读取在 T1 采样的 X 和 Y 以及在 T2 采样的 Z。*

## 4.2 了解加速度数据

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。

### 4.2.1 数据对齐

加速度数据表示为 16 位数，向左对齐（低四位无效）。分辨率取决于选择的功耗模式。

### 4.2.2 大小端序选择

LIS2DH12 允许交换加速度寄存器低位部分和高位部分的内容（即交换 OUT_X_L 与 OUT_X_H 的内容），以便符合小端和大端数据表示法的要求。

“小端模式”表示数字的低位字节存储在存储器的最低地址中，高位字节存储在最高地址中。（小端模式优先）。该模式相当于 CTRL_REG4 中的 BLE 位复位为 0（默认配置）。

相反，“大端模式”表示数字的高位字节存储在存储器的最低地址中，低位字节存储在最高地址中。

只有在高分辨率模式下才能激活 BLE 功能。

### 4.2.3 加速度数据示例

表 9. 输出数据寄存器内容 vs. 加速度（FS = ±2 *g*，高分辨率模式） 提供的几个基本示例中，会在器件受给定加速度影响的情况下读取数据寄存器中的数据。表中列出的数值均假定器件已进行准确校准（也就是没有偏移、没有增益误差），并且会真实显示 BLE 位的影响。

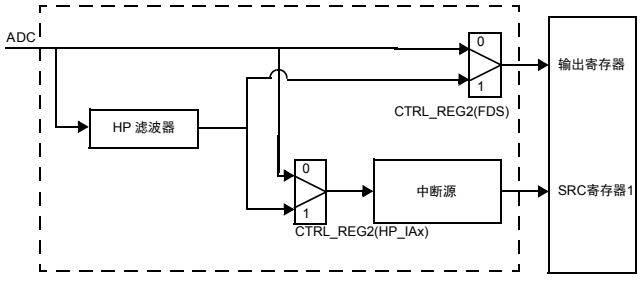
**表 9. 输出数据寄存器内容 vs. 加速度（FS = ±2 *g*，高分辨率模式）**

| 加速度值 | BLE = 0 | | BLE = 1 | |
|---|---|---|---|---|
| | 寄存器地址 | | | |
| | 28h | 29h | 28h | 29h |
| 0 *g* | 00h | 00h | 00h | 00h |
| 350 m*g* | E0h | 15h | 15h | E0h |
| 1 *g* | 00h | 40h | 40h | 00h |
| -350 m*g* | 20h | EAh | EAh | 20h |
| -1 *g* | 00h | C0h | C0h | 00h |

## 4.3 高通滤波器

LIS2DH12 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 4. 高通滤波器连接框图中所示，通过配置 CTRL_REG2 的 FDS、HP_IA1 和 HP_IA2 位，可以将滤波器独立应用于输出数据和/或中断数据。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。

**图 4. 高通滤波器连接框图**




### 4.3.1 滤波器配置

参照表 10. 高通滤波器模式配置，高通滤波器可具有两种工作模式：

**表 10. 高通滤波器模式配置**

| HPM1 | HPM0 | 高通滤波器模式 |
|---|---|---|
| 0 | 0 | 正常模式（通过读取 REFERENCE（26h）复位） |
| 0 | 1 | 滤波参考信号 |

| HPM1 | HPM0 | 高通滤波器模式 |
|---|---|---|
| 1 | 0 | 正常模式 |
| 1 | 1 | 中断事件自动复位 |

高通滤波器的带宽取决于所选 ODR 以及 CTRL_REG2 的 HPCFx 位的设置。高通滤波器截止频率（$f_t$）如表 11. 低功耗模式 - 高通滤波器截止频率[Hz]中所示。

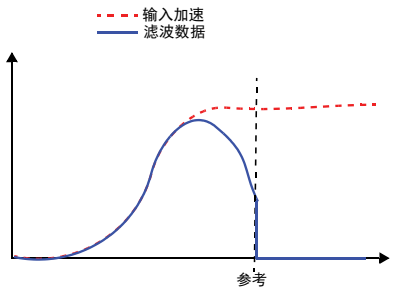
**表 11. 低功耗模式 - 高通滤波器截止频率[Hz]**

| HPCF[2:1] | $f_t$ [Hz] @1 Hz | $f_t$ [Hz] @10 Hz | $f_t$ [Hz] @25 Hz | $f_t$ [Hz] @50 Hz | $f_t$ [Hz] @100 Hz | $f_t$ [Hz] @200 Hz | $f_t$ [Hz] @400 Hz | $f_t$ [Hz] @1.6 kHz | $f_t$ [Hz] @5 kHz |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 0.02 | 0.2 | 0.5 | 1 | 2 | 4 | 8 | 32 | 100 |
| 01 | 0.008 | 0.08 | 0.2 | 0.5 | 1 | 2 | 4 | 16 | 50 |
| 10 | 0.004 | 0.04 | 0.1 | 0.2 | 0.5 | 1 | 2 | 8 | 25 |
| 11 | 0.002 | 0.02 | 0.05 | 0.1 | 0.2 | 0.5 | 1 | 4 | 12 |

#### 4.3.1.1 *正常模式*

在该配置下，可通过读取 REFERENCE（26h）复位高通滤波器，从而立即删除加速度的 DC 分量。

**图 5. 读取 REFERENCE**



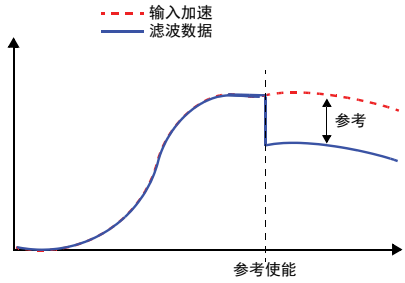

#### 4.3.1.2 *参考模式*

在该配置下，输出数据会计算为输入加速度与 REFERENCE（26h）内容之差。该寄存器表示为 2 的补码形式，这些 7 位寄存器的 1 LSB 的值取决于所选满量程(表 12. 参考模式 LSB 值)。

**表 12. 参考模式 LSB 值**

| 满量程 | 参考模式 LSB 值(mg) |
|---|---|
| ±2 *g* | ~16 |
| ±4 *g* | ~31 |
| ±8 *g* | ~63 |
| ±16 *g* | ~127 |

图 6. 参考模式

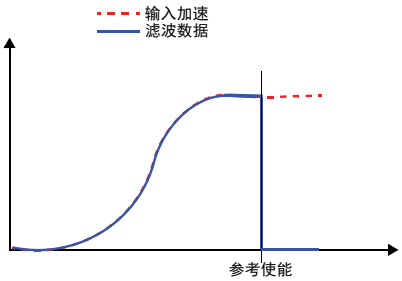



#### 4.3.1.3 *自动复位*

在该配置下，发生配置的中断事件时，滤波器会自动复位。但会立即使用 REFERENCE（26h）将滤波器置位。

*请注意：用于复位滤波器的 XYZ 数据集是中断后的数据集。*

图 7. 自动复位



# 5 中断生成

LIS2DH12 中断信号可用作自由落体、唤醒、6D 和 4D 定向检测以及点击检测。这些信号可驱动到两个中断引脚（INT1 和 INT2）。

## 5.1 中断引脚配置

器件提供的两个引脚可激活为生成数据就绪信号或中断信号。引脚功能是通过 CTRL_REG3(22h)和 CTRL_REG6(25h)选择的。

**表 13. CTRL_REG3 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| I1_CLICK | I1_IA1 | I1_IA2 | I1_ZYXDA | 0(1) | I1_WTM | I1_OVERRUN | -- |

1. *为了设备的正确运行，此位必须置为“0”。*

**表 14. CTRL_REG3 说明**

| | |
|---|---|
| I1_CLICK | INT1 上的 CLICK 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_IA1 | INT1 上的 IA1 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_IA2 | INT1 上的 IA2 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_ZYXDA | INT1 上的 ZYXDA 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_WTM | INT1 上的 FIFO 水印中断。默认值：0<br>（0：禁用；1：启用） |
| I1_OVERRUN | INT1 上的 FIFO 上溢中断。默认值：0<br>（0：禁用；1：启用） |

**表 15. CTRL_REG6 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| I2_CLICK | I2_IA1 | I2_IA2 | I2_BOOT | I2_ACT | - - | INT_POLARITY | - |

**表 16. CTRL_REG6 说明**

| | |
|---|---|
| I2_CLICK | INT2 上的 CLICK 中断。默认值：0<br>（0：禁用；1：启用） |
| I2_IA1 | 使能 INT2 引脚上的中断 1 功能。默认值：0<br>（0：功能禁用；1：功能使能） |
| I2_IA2 | 使能 INT2 引脚上的中断 2 功能。默认值：0<br>（0：功能禁用；1：功能使能） |
| I2_BOOT | 使能 INT2 引脚上的启动。默认值：0<br>（0：禁用；1：启用） |

| | |
|---|---|
| I2_ACT | 使能 INT2 引脚上的活动中断。默认值：0<br>（0：禁用；1：启用） |
| INT_POLARITY | INT1 和 INT2 引脚的极性。默认值：0<br>（0：高电平有效；1：低电平有效） |

# 6 惯性中断

LIS2DH12 可提供两个惯性中断信号，并可通过多种方式对这两个信号进行定制化。中断生成行为中涉及到的寄存器是 INTx_CFG、INTx_THS 和 INTx_DURATION。

表 17. 中断模式配置

| AOI | 6D | 中断模式 |
|---|---|---|
| 0 | 0 | 中断事件的 OR（或运算）组合 |
| 0 | 1 | 6 方向运动识别 |
| 1 | 0 | 中断事件的 AND（与运算）组合 |
| 1 | 1 | 6 方向位置识别 |

中断条件满足时，会生成中断信号，通过读取 INTx_SRC 寄存器，可以了解发生了什么情况。

## 6.1 持续时间

持续时间寄存器的内容会设置要识别的中断时间的最短持续时间。持续时间步数和最大值取决于选择的 ODR。
持续时间的测量单位为 N/ODR，其中，N 是持续时间寄存器的内容。

表 18. 正常模式下的持续时间 LSB 值

| ODR（Hz） | 持续时间 LSB 值（ms） |
|---|---|
| 1 | 1000 |
| 10 | 100 |
| 25 | 40 |
| 50 | 20 |
| 100 | 10 |
| 200 | 5 |
| 400 | 2.5 |
| 1600 | 0.6 |
| 1344 | 0.744 |
| 5376 | 0.186 |

## 6.2 阈值

阈值寄存器定义了中断生成电路使用的参考加速度。这些 7 位寄存器的 1 LSB 的值取决于所选满量程和功耗模式（请参考 LIS2DH12 数据表的“表 4：机械特性”）。

## 6.3 自由落体和唤醒中断

LIS2DH12 中断信号可当做自由落体和唤醒中断。中断条件满足时，会生成中断信号，通过读取 INTx_SRC 寄存器，可以了解发生了什么情况。

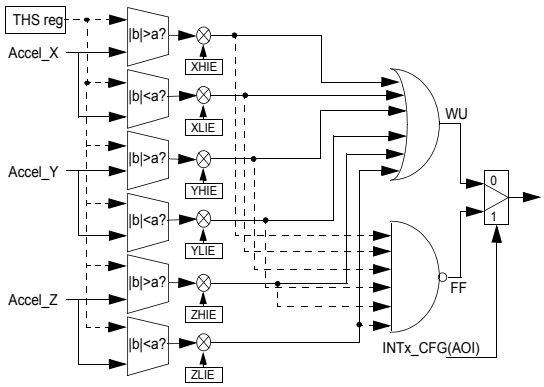
自由落体信号(FF)和唤醒信号(WU)中断生成块在图 8. 自由落体、唤醒中断发生器中表示。

FF 或 WU 中断生成是通过 INTx_CFG 寄存器中的 AOI 位选择的。如果 AOI 位为“0”，来自轴（通过 INTx_CFG 寄存器使能）比较器的信号会输入到逻辑 OR 中。在这种情况下，当至少有一个已使能轴超出写入到 INTx_THS 寄存器模块中的阈值时，会生成中断。如果 AOI 位为“1”，来自比较器的信号会进入“NAND”端口。在这种情况下，仅当所有已使能轴都超过写入到 INTx_THS 寄存器中的阈值时，才会生成中断信号。

CTRL_REG5 的 LIR_INTx 位可决定中断请求是否必须锁存。如果 LIR_INTx 位为“0”，当中断条件满足时，中断信号会变为高电平，如果中断条件不再满足，中断信号会立即恢复低电平。否则，如果 LIR_INTx 位为“1”，如果中断条件适用，即使条件恢复为非中断状态，中断信号也会保持高电平，直至对 INTx_SRC 寄存器执行读取操作。

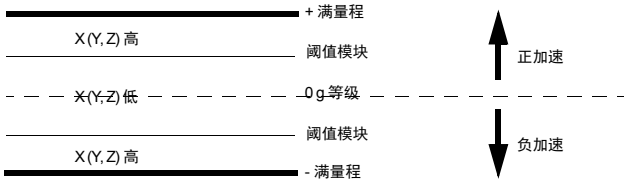
INTx_CFG 寄存器的 ZHIE、ZLIE、YHIE、YLIE、XHIE 和 XLIE 位可决定必须对哪一条轴执行中断决策、并可决定必须沿哪一方向传递阈值才能生成中断请求。

图 8. 自由落体、唤醒中断发生器




系统检测任何自由落体或惯性唤醒事件所使用的阈值模块是由 INTx_THS 寄存器定义的。阈值表示为 7 位无符号数字，并且绕零重力水平对称。如果 X (Y, Z)通道的无符号加速度值大于 INTx_THS，XH (YH, ZH)为真。同样，如果 X (Y, Z)通道的无符号加速度值小于 INTx_THS，XL (YL, ZL)低电平为真。更多详情，请参阅图 9. FF_WU_CFG 高电平和低电平。

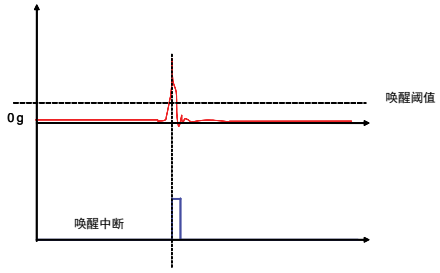
图 9. FF_WU_CFG 高电平和低电平




### 6.3.1 惯性唤醒

唤醒中断是指 INTx_CTRL 寄存器的特定配置，该配置允许在已配置轴的加速度超过定义的阈值(图 10. 惯性唤醒中断)时生成中断。

图 10. 惯性唤醒中断




### 6.3.2 不使用高通滤波器

本段介绍的基本算法演示了惯性唤醒功能的实际应用。下列代码会将器件配置为可识别出沿 X 或 Y 轴方向的绝对加速度超过预设阈值的情况。

（本例中使用的是 250 mg）。触发中断的事件会锁存在设备内，会使用 INT1 引脚指示发生该事件。

| | | |
|---|---|---|
| 1. | 将 57h 写入 CTRL_REG1 | // 打开传感器并使能 X、Y 和 Z<br>// ODR = 100 Hz |
| 2. | 将 00h 写入 CTRL_REG2 | // 关闭高通滤波器 |
| 3. | 将 40h 写入 CTRL_REG3 | // 中断活动 1 挂载到 INT1 引脚上 |
| 4. | 将 00h 写入 CTRL_REG4 | // FS = ±2 *g* |
| 5. | 将 08h 写入 CTRL_REG5 | // 中断 1 引脚已锁存 |
| 6. | 将 10h 写入 INT1_THS | // 阈值 = 250 mg |
| 7. | 将 00h 写入 INT1_DURATION | // 持续时间 = 0 |
| 8. | 将 0Ah 写入 INT1_CFG | // 使能 XH 和 YH 中断生成 |
| 9. | 轮询 INT1 焊盘；如果 INT1=0，则转至 8 | // 轮询 DRDY/INT1 引脚等待<br>// 唤醒事件 |
| 10. | 读 INT1_SRC | // 返回触发了中断<br>// 的事件 |
| 11. | （发生了唤醒事件；在此插入您的代码） | // 事件处理 |
| 12. | 进入 8 | |

### 6.3.3 使用高通滤波器

以下代码中的基本例程展示了对已进行高通滤波的数据执行的惯性唤醒功能的实际应用。器件被配置为可识别出施加到 X、Y 或 Z 轴的加速度的高频分量超过预设阈值（本例中使用的阈值为 250 mg）的情况。
触发中断的事件会锁存在设备内，会使用 INT1 引脚指示发生该事件。

| | | |
|---|---|---|
| 1. | 将 57h 写入 CTRL_REG1 | // 启动传感器，使能 X、Y 和 Z<br>// ODR = 100 Hz |
| 2. | 将 09h 写入 CTRL_REG2 | // 中断活动 1 已使能高通滤波器 |
| 3. | 将 40h 写入 CTRL_REG3 | // 中断活动 1 挂载到 INT1 引脚上 |
| 4. | 将 00h 写入 CTRL_REG4 | // FS = ±2 *g* |
| 5. | 将 08h 写入 CTRL_REG5 | // 中断 1 引脚已锁存 |
| 6. | 将 10h 写入 INT1_THS | // 阈值 = 250 mg |
| 7. | 将 00h 写入 INT1_DURATION | // 持续时间 = 0 |
| 8. | 读取 REFERENCE | // 进行虚拟读取，将高通滤波器强制设为<br>// 当前加速度值<br>//（也就是设置参考加速度/倾斜值） |
| 9. | 将 2Ah 写入 INT1_CFG | // 配置所需唤醒事件 |
| 10. | 轮询 INT1 焊盘；如果 INT1=0，则转至 9 | // 轮询 INT1 引脚等待<br>// 唤醒事件 |
| 11. | （发生了唤醒事件；在此插入您的代码） | // 事件处理 |
| 12. | 读 INT1_SRC | // 返回触发了中断<br>// 中断并清除中断 |
| 13. | （在此插入您的代码） | // 事件处理 |
| 14. | 转至 9 | |

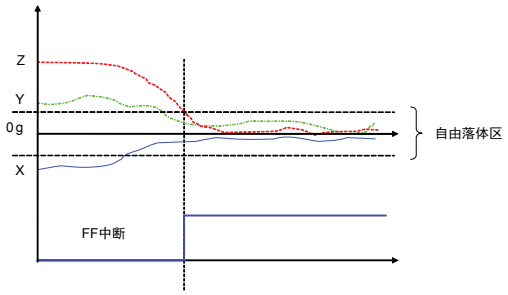
在第 8 步中，会对 REFERENCE 寄存器执行虚拟读取，以便设置器件执行阈值比较时所参照的当前/参考加速度/倾斜状态。

可根据需要随时执行虚拟读取，以便将器件的方向/倾斜设为参考状态，无需等待滤波器稳定下来。

## 6.4 自由落体检测

自由落体检测是指利用 INTx_CTRL 寄存器的特定配置来识别器件是否在进行自由落体运动：沿所有轴测量的加速度均变为零。在实际情况下“自由落体区”定义为零重力水平附近，在该区域中，所有加速度都足够小，可生成中断 (图 11. 自由落体中断)。

**图 11. 自由落体中断**




本段介绍了使用自由落体检测的基础知识。还特别介绍了以下将器件配置为检测自由落体事件并发信号指示此类事件的软件例程：

| | | |
|---|---|---|
| 1. | 将 57h 写入 CTRL_REG1 | // 启动传感器，使能 X、Y 和 Z<br>// ODR = 100 Hz |
| 2. | 将 00h 写入 CTRL_REG2 | // 关闭高通滤波器 |
| 3. | 将 40h 写入 CTRL_REG3 | // 中断活动 1 挂载到 INT1 引脚上 |
| 4. | 将 00h 写入 CTRL_REG4 | // FS = ±2 *g* |
| 5. | 将 08h 写入 CTRL_REG5 | // 中断 1 引脚已锁存 |
| 6. | 将 16h 写入 INT1_THS | // 将自由落体阈值设为 350 mg |
| 7. | 将 03h 写入 INT1_DURATION | // 设置最短事件持续时间 |
| 8. | 将 95h 写入 INT1_CFG | // 配置自由落体识别 |
| 9. | 轮询 INT1 焊盘；如果 INT1=0，则转至 10 | // 轮询等待自由落体事件的 INT1 引脚 |
| 10. | （发生了自由落体事件；在此插入您的代码） | // 事件处理 |
| 11. | 读取 INT1_SRC 寄存器 | // 清除中断请求 |
| 12. | 转至 9 | |

示例代码利用设定为 350 mg 的阈值进行自由落体识别，并通过硬件信号 INT1 通知自由落体事件。在第 7 步，INT1_DURATION 寄存器像这样进行了配置，从而可忽略短于 3/DR = 3/100 ~= 30 ms 的事件，以避免假检测现象的发生。

发生自由落体事件后，对 INT1_SRC 寄存器的读取操作会清空请求，器件会准备好识别其他事件。

# 7 6D/4D 定向检测

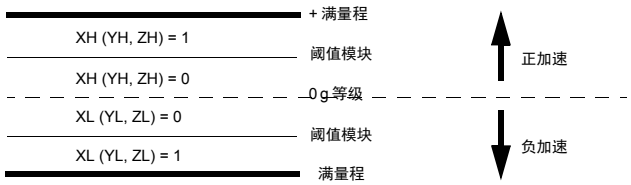
LIS2DH12 的高级功能可检测器件在空间中的方向，从而可轻松为手持设备实施节能程序，并可实现图像自动旋转。

## 7.1 6D 定向检测

6D 定向方向功能可通过 INTx_CFG 寄存器的 AOI 和 6D 位启用。配置为实现 6D 功能时，INTx_SRC 的 ZH、ZL、YH、YL、XH 和 XL 位会提供关于加速度值以及加速度信号的相关信息，当加速度值大于阈值时，会生成中断。更具体地说：

- 当感应到的加速度大于正方向阈值时，ZH (YH, XH)为 1
- 当感应到的加速度大于负方向阈值时，ZL (YL, XL)为 1

**图 12. ZH、ZL、YH、YL、XH 和 XL 特性**




6D 方向功能有两种可行的配置：

- 6D 运动识别：在该配置下，当器件从一个方向（已知或未知方向）移动到另一已知方向时，会生成中断。中断的有效持续时间仅为 1/ODR。
- 6D 位置识别：在该配置下，当器件在已知方向处于稳定状态时，会生成中断。只要位置保持不变（图 13. 6D 运动与 6D 位置对比, (a) 和(b)），中断就有效。

参考图 13. 6D 运动与 6D 位置对比，6D 运动行显示了当器件配置为在 X 和 Y 轴上实现 6D 运动识别(INTx_CFG = 0x4Ah)时中断的行为，而 6D 位置行则显示了当器件配置为在 X 和 Y 轴上实现 6D 位置识别(INT1_CFG = 0xCAh)时中断的行为。INT1_THS 设为 0x21。

参考图 14. 6D 识别位置，器件配置为在 X、Y 和 Z 轴上实现 6D 位置功能。 表 19. 6D 定位下的 INT1_SRC 寄存器显示的是每个位置的 INT1_SRC 寄存器的内容。

**图 13. 6D 运动与 6D 位置对比**

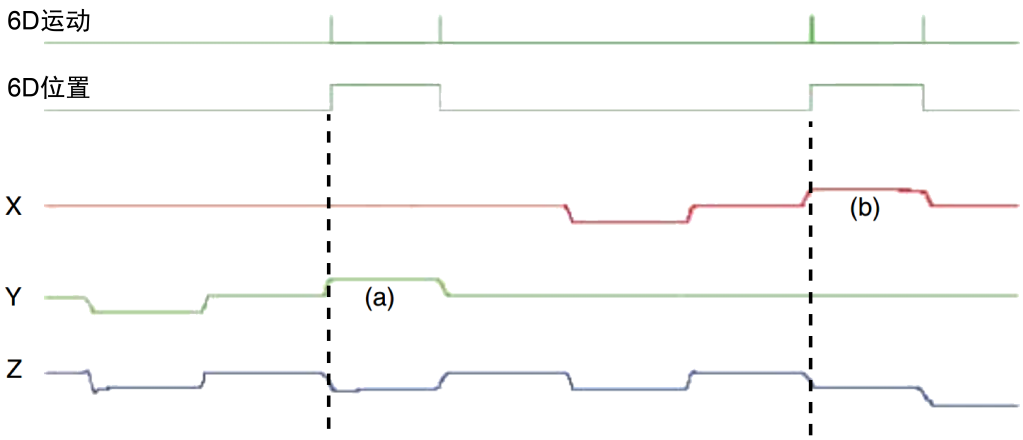


图 14. 6D 识别位置

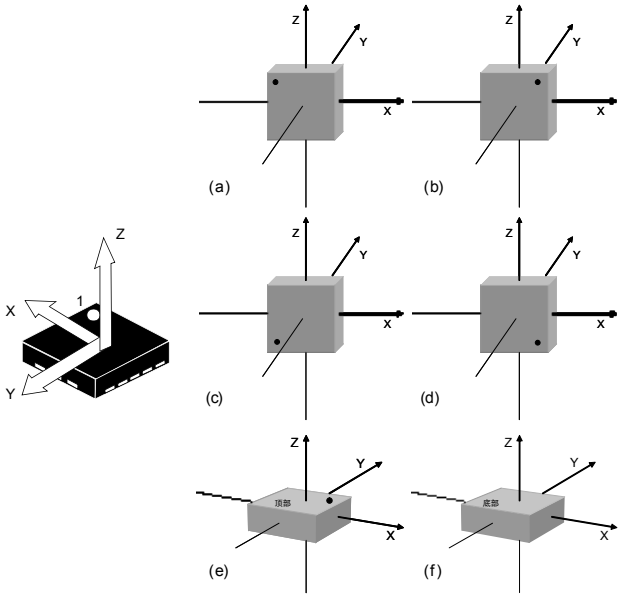



表 19. 6D 定位下的 INT1_SRC 寄存器

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

## 7.2 4D 定向方向

4D 检测是 6D 功能的子集，它被专门定义来进行手持设备中的纵向和横向计算。当 INTx_CFG 的 6D 位设为 1 时，可通过将 CTRL_REG5 的 D4D_INTx 位置 1 来启用此功能。这种配置下，Z 轴位置检测被禁用，因此位置识别减少为表 19. 6D 定位下的 INT1_SRC 寄存器的(a)、(b)、(c)和(d)的情形。

# 8 单击和双击识别

借助 LIS2DH12 中的单击和双击识别功能，无需载入软件便可创建人机界面。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。

如果传感器受到单个输入刺激，则会在惯性引脚 INT1 和/或 INT2 上生成中断请求。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。

此功能完全可通过用户编程进行设定，用户可在程序中通过 Section 8.3 寄存器说明中介绍的专用寄存器组设定刺激的预期幅度和时间。

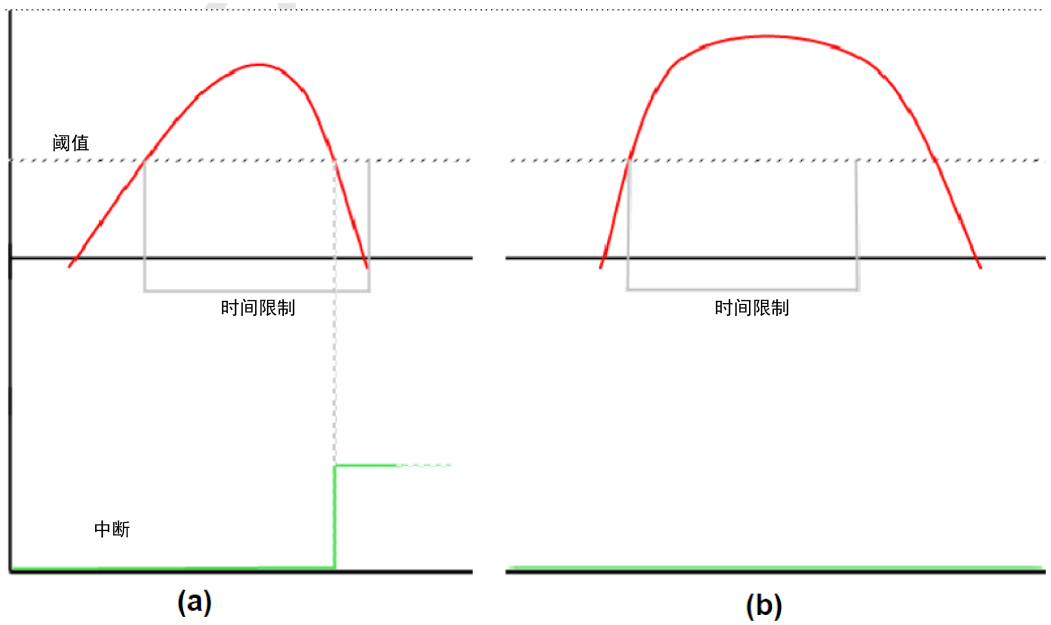
用于单击和双击识别的推荐加速度计 ODR 为 400 Hz 或更高。

## 8.1 单击

如果器件配置为实现单击事件识别，如果所选通道上的输入加速度超过设定的阈值、并且在由 TIME_LIMIT 寄存器定义的时窗内恢复至阈值以下，则会生成中断，

如果 CLICK_THS 寄存器的 LIR_Click 位未置 1，中断会保持高电平，并会在延迟窗口的持续时间内一直处于高电平状态。如果 LIR_Click 位已置 1，中断会保持高电平，直至 CLICK_SRC 寄存器被读取。

**图 15. 使用非锁存中断的单击事件**




在图 15. 使用非锁存中断的单击事件(a)中，已识别出点击事件，而在图 15. 使用非锁存中断的单击事件(b)中还未识别出点击，因为加速度在 TIME_LIMIT 到期后降至阈值以下。

## 8.2 双击

如果器件配置为实现双击事件检测，当第一次点击后识别出第二次点击时，会生成中断。仅当事件满足延迟寄存器和窗口寄存器定义的规则时，才会识别第二次点击。

特别是在已识别了第一次点击后，第二次点击检测程序会延迟进行，延迟的时间间隔是延迟寄存器定义的。这意味着在识别出第一次点击后，仅当输入加速度在延迟窗口后、但在窗口到期(图 16. 单击和双击识别 (a))前超过阈值、或者加速度在延迟到期( (b))后超过阈值的情况下才会开始执行第二次点击检测程序。

第二次点击检测程序启动后，会采用与识别第一次点击时一样的程序识别第二次点击：加速度必须在 TIME_LIMIT 到期前恢复为阈值以下。

请务必正确定义延迟窗口设置（TIME_LATENCY 寄存器），以避免因输入信号虚假反弹出现不需要的点击。

图 16. 单击和双击识别

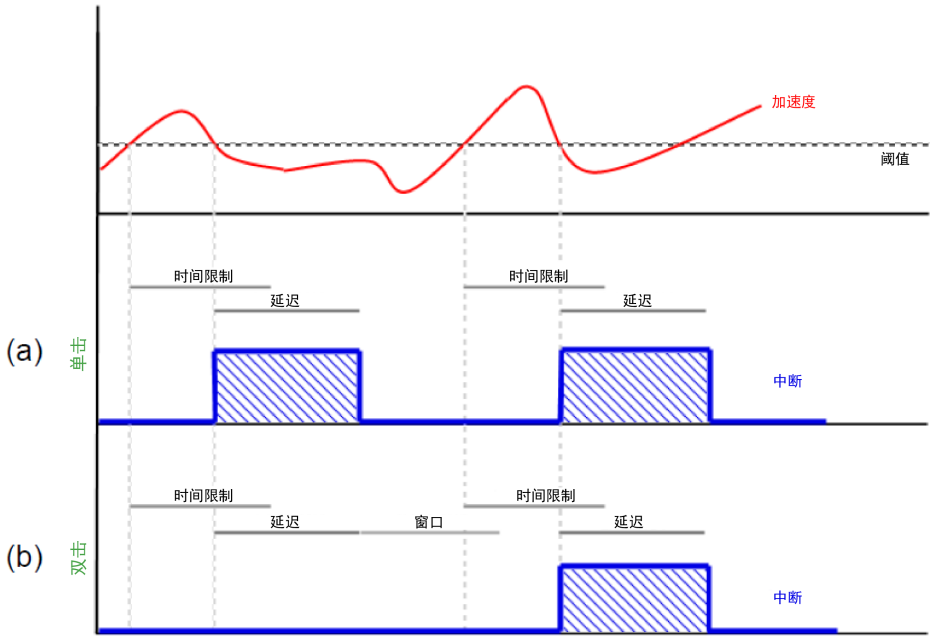
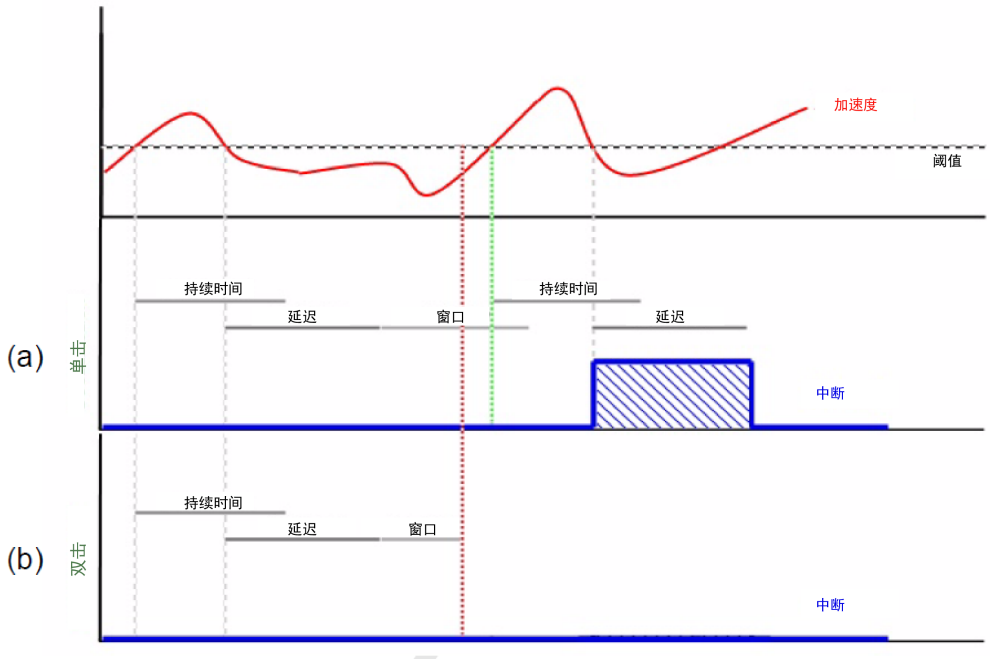



图 16. 单击和双击识别 通过图形介绍了单击事件(a)和双击事件(b)。器件能够通过将 CLICK_CFG 寄存器的设置从单击识别改为双击识别的方式区分(a)和(b)。

图 17. 双击识别




在图 17. 双击识别(a)中，已正确识别出双击事件，而在图 17. 双击识别(b)中还未生成中断，因为输入加速度在窗口时间间隔到期后超过阈值。

## 8.3 寄存器说明

### 8.3.1 CLICK_CFG (38h)

表 20. CLICK_CFG 寄存器

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | - | ZD | ZS | YD | YS | XD | XS |

表 21. CLICK_CFG 说明

| | |
|---|---|
| ZD | 在 Z 轴上启用中断双连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |
| ZS | 在 Z 轴上启用中断单连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |
| YD | 在 Y 轴上启用中断双连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |
| YS | 在 Y 轴上启用中断单连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |
| XD | 在 X 轴上启用中断双连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |
| XS | 在 X 轴上启用中断单连击。默认值：0<br>（0：禁用中断请求；<br>1：对高于预设阈值的测得加速度启用中断请求） |

表 22. 真值表

| DZ / DY / DX | SZ / Y / X | 点击输出 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 单 |
| 1 | 0 | 双 |
| 1 | 1 | 双 |

### 8.3.2 CLICK_SRC (39h)

表 23. CLICK_SRC 寄存器

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | IA | DCLICK | SCLICK | 符号 | Z | 是 | X |

表 24. CLICK_SRC 说明

| | |
|---|---|
| IA | 中断有效。默认值：0<br>（0：未生成中断；1：已生成一个或多个中断） |

| | |
|---|---|
| DCLICK | 双击使能。默认值：0<br>（0：双击检测禁用，1：双击检测使能） |
| SCLICK | 单击使能。默认值：0<br>（0：单击检测禁用，1：单击检测使能） |
| 符号 | 点击符号。<br>（0：正检测，1：负检测） |
| Z | Z 轴点击检测。默认值：0<br>（0：无中断，1：发生了 Z 轴置为高电平的事件） |
| 是 | Y 轴点击检测。默认值：0<br>（0：无中断，1：发生了 Y 轴置为高电平的事件） |
| X | X 轴点击检测。默认值：0<br>（0：无中断，1：发生了 X 轴置为高电平的事件） |

### 8.3.3 CLICK_THS (3Ah)

表 25. **CLICK_THS 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| LIR_Click | Ths6 | Ths5 | Ths4 | Ths3 | Ths2 | Ths1 | Ths0 |

表 26. **CLICK_THS 说明**

| | |
|---|---|
| LIR_Click | 如果 LIR_Click 位未置 1，中断会保持高电平，并会在延迟窗口的持续时间内一直处于高电平状态。<br>如果 LIR_Click 位已置 1，中断会保持高电平，直至 CLICK_SRC (39h)被读取。 |
| Ths[6:0] | 点击阈值。默认值：000 0000 |

1 LSB = 满量程/128。
Ths6 到 Ths0 定义了系统启动点击检测程序所使用的阈值。阈值表示为 6 位无符号数。

### 8.3.4 TIME_LIMIT (3Bh)

表 27. **TIME_LIMIT 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | TLI6 | TLI5 | TLI4 | TLI3 | TLI2 | TLI1 | TLI0 |

表 28. **TIME_LIMIT 寄存器说明**

| | |
|---|---|
| TLI7-TLI0 | 点击时间限制。默认值：000 0000 |

1 LSB = 1/ODR。
TLI7 到 TLI0 定义了启动点击检测程序（选定通道上的加速度超过设定的阈值）与加速度变回阈值以下所经过的最大时间间隔。

### 8.3.5 TIME_LATENCY (3Ch)

表 29. **TIME_LATENCY 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| TLA7 | TLA6 | TLA5 | TLA4 | TLA3 | TLA2 | TLA1 | TLA0 |

表 30. **TIME_LATENCY 说明**

| | |
|---|---|
| TLA7-TLA0 | 点击时间延迟。默认值：000 0000 |

1 LSB = 1/ODR。
如果器件配置为实现双击检测，TLA7 到 TLA0 定义了在第一次点击检测后开始的时间间隔，在这段时间内会禁用点击检测程序。

### 8.3.6 时窗(3Dh)

表 31. **TIME_WINDOW 说明**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| TW7 | TW6 | TW5 | TW4 | TW3 | TW2 | TW1 | TW0 |

表 32. **TIME_WINDOW 说明**

| | |
|---|---|
| TW7-TW0 | 点击时窗 |

1 LSB = 1/ODR。
如果器件配置为实现双击检测，TW7 到 TW0 定义了延迟间隔结束后可经过的最大时间间隔，在这段时间内可开始点击检测程序。

### 8.3.7 CTRL_REG3 [中断 CTRL 寄存器] (22h)

表 33. **CTRL_REG3 寄存器**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| I1_CLICK | I1_IA1 | I1_IA2 | I1_ZYXDA | 0(1) | I1_WTM | I1_OVERRUN | -- |

1. *为了设备的正确运行，此位必须置为“0”。*

表 34. **CTRL_REG3 说明**

| | |
|---|---|
| I1_CLICK | INT1 上的 CLICK 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_IA1 | INT1 上的 IA1 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_IA2 | INT1 上的 IA2 中断。默认值：0<br>（0：禁用；1：启用） |
| I1_ZYXDA | INT1 上的 ZYXDA 中断。默认值：0<br>（0：禁用；1：启用） |

| | |
|---|---|
| I1_WTM | INT1 上的 FIFO 水印中断。默认值：0<br>（0：禁用；1：启用） |
| I1_OVERRUN | INT1 上的 FIFO 上溢中断。默认值：0<br>（0：禁用；1：启用） |

## 8.4 示例

下图显示的是不同条件下的点击中断生成情况。截图是在运行演示套件 GUI 接口的电脑上获取的，ODR 设为 400 Hz，满量程设为 4 *g*。LIS2DH12 寄存器的内容已通过软件界面的专用面板进行修改，用户可在该面板中对嵌入了点击的功能的所有不同设置和特性进行评估。下例中，仅启用了 X 轴生成点击中断。

### 8.4.1 调整 TIME_LIMIT

图 18. 短时间限制 显示的是 TIME_LIMIT = 01h (2.5 ms)时执行的采集。使用该设置时，单击识别窗口较短，通常加速度不会及时恢复为阈值以下。

图 19. 长时间限制中显示的是 TIME_LIMIT = 33h (127 ms)时完成的采集。使用该设置时，单击识别窗口较长，事件更容易识别。

**图 18. 短时间限制**

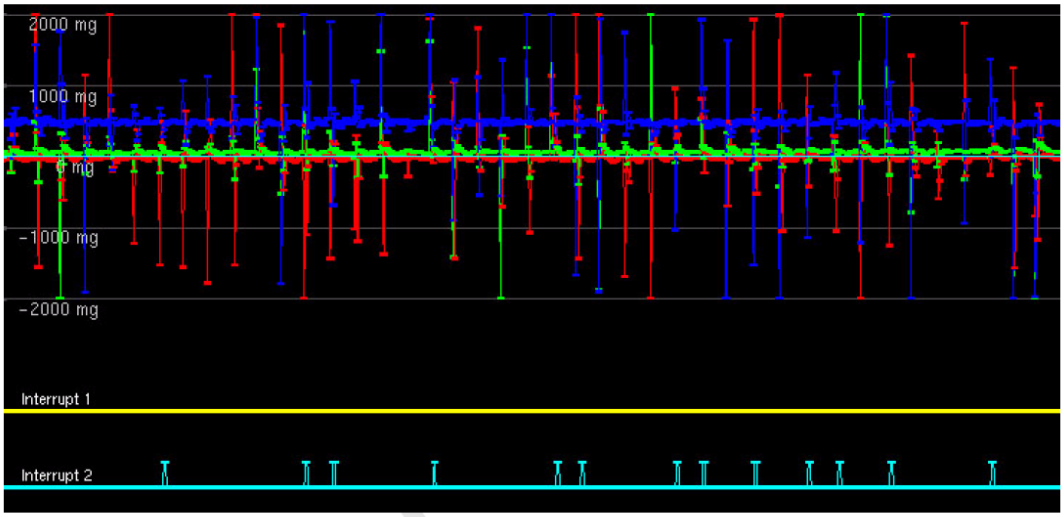
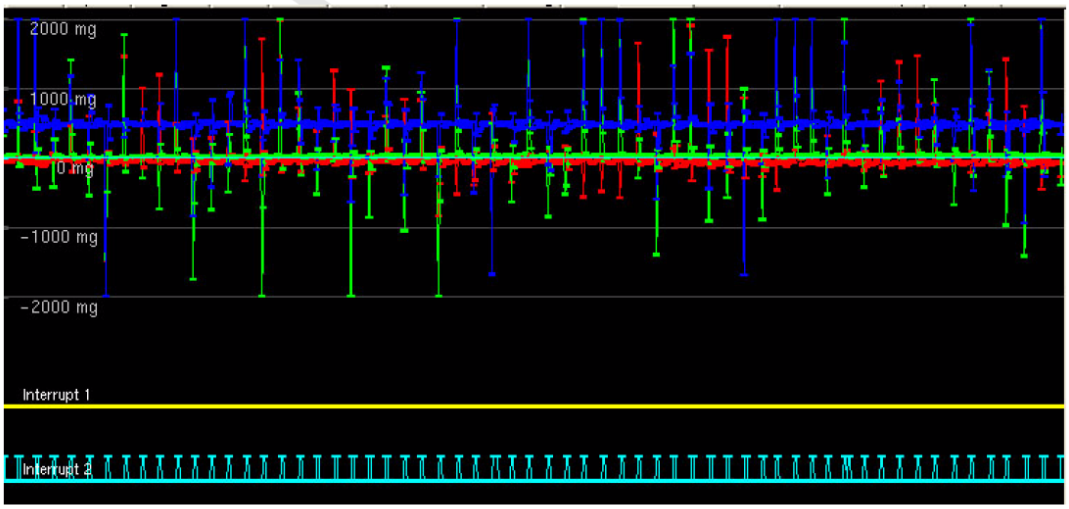


图 19. 长时间限制




### 8.4.2 调整 TIME_LATENCY

图 20. 短延迟 显示的是 TIME_LATENCY =15h (52 ms)时进行的采集。使用该设置时，器件几乎会将每个加速度尖峰识别为点击操作。

图 21. 长延迟中显示的是 TIME_LATENCY = FFh (637 ms)时执行的采集。使用该设置时，器件会将每两个尖峰中的一个识别为点击操作。

图 20. 短延迟

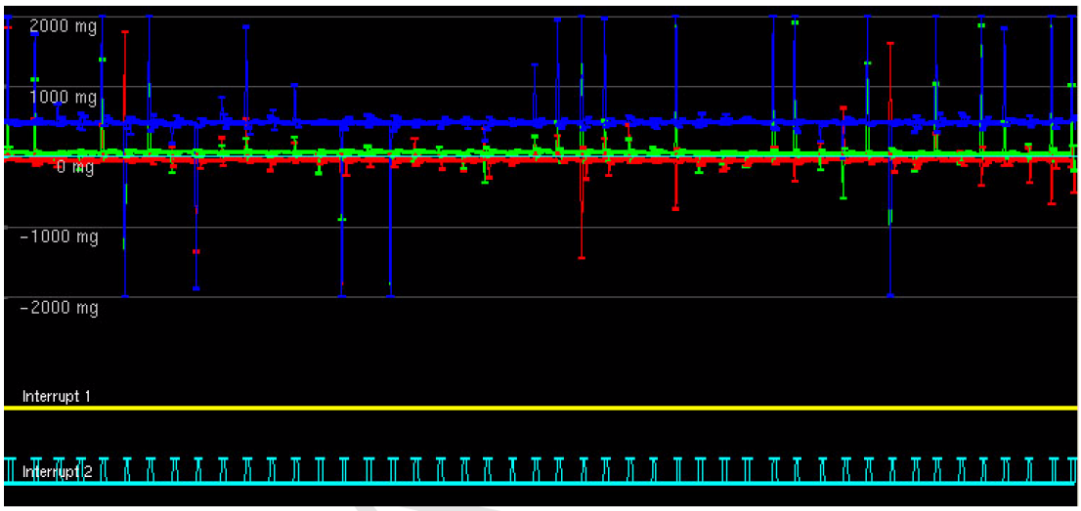
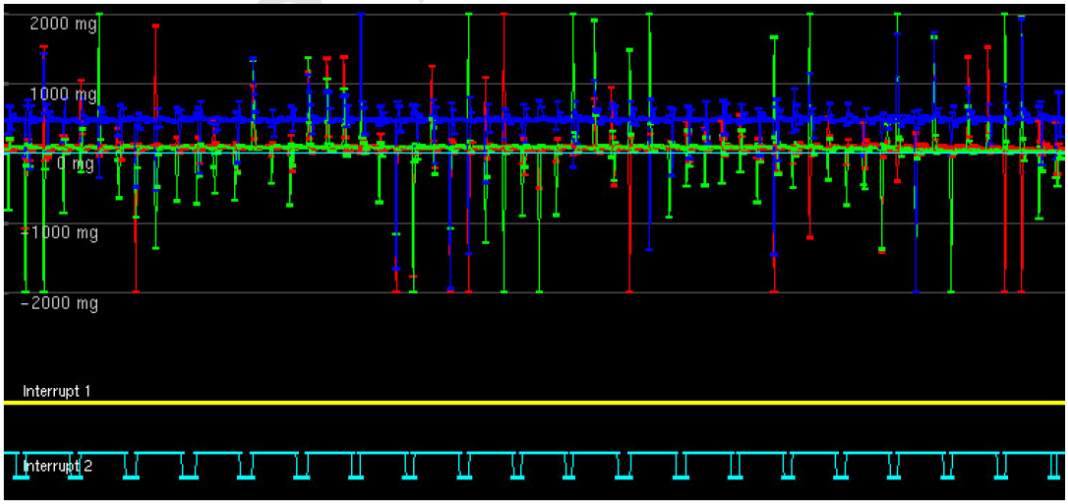


图 21. 长延迟




### 8.4.3 调整 TIME_WINDOW

进行双击识别时，TIME_LATENCY + TIME_WINDOW 定义了将两次连续点击识别为双击事件的最大间隔时间。固定延迟可避免信号虚假反弹，可以像在电脑上调整鼠标属性的“双击速度”设置一样调整 TIME_WINDOW。

图 22. 短窗口 显示的是 TIME_WINDOW = 42h (165 ms)时进行的采集。使用该设置时，加速度的两个连续峰值间隔越远，第二个峰值出现在窗口之外。

图 23. 长窗口中显示的是 TIME_WINDOW = FFh (637 ms)时执行的采集。使用该设置时，器件会在第二个加速度峰值后正确生成双击中断。

图 22. 短窗口

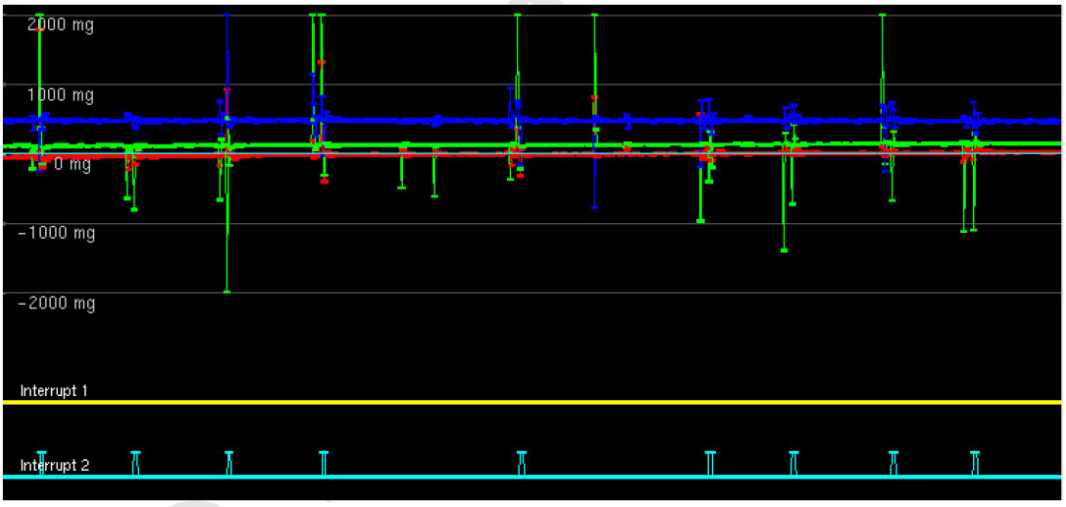
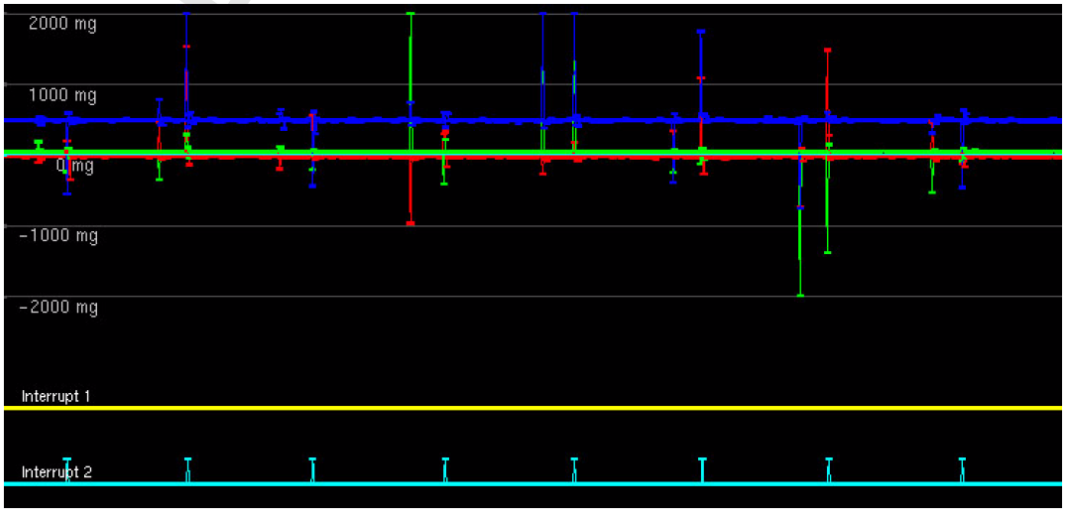


图 23. 长窗口



# 9 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，LIS2DH12 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 缓冲区可在四种不同模式下工作，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、Stream 模式和 Stream-to-FIFO 模式。

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

## 9.1 FIFO 描述

FIFO 缓冲区最多能为每条通道存储 32 个加速度采样；数据采用左对齐 16 位 2 的补码形式存储。分辨率取决于选择的功耗模式。

数据样本集合由 6 个字节（X_low、X_high、Y_low、Y_high、Z_low 和 Z_high）组成，它们会以选定的输出数据率（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 35. FIFO 缓冲区填满示例（存储第 32 个采样集）**

| 输出寄存器 | 0x28h | 0x29h | 0x2Ah | 0x2Bh | 0x2Ch | 0x2Dh |
|---|---|---|---|---|---|---|
| | X_low(0) | X_high(0) | Y_low(0) | Y_high(0) | Z_low(0) | Z_high(0) |
| FIFO 索引 | FIFO 样本集合 | | | | | |
| FIFO（0） | X_low(0) | X_high(0) | Y_low(0) | Y_high(0) | Z_low(0) | Z_high(0) |
| FIFO（1） | X_low(1) | X_high(1) | Y_low(1) | Y_high(1) | Z_low(1) | Z_high(1) |
| FIFO（2） | X_low(2) | X_high(2) | Y_low(2) | Y_high(2) | Z_low(2) | Z_high(2) |
| FIFO（3） | X_low(3) | X_high(3) | Y_low(3) | Y_high(3) | Z_low(3) | Z_high(3) |
| ... | ... | ... | ... | ... | ... | ... |
| FIFO（30） | X_low(30) | X_high(30) | Y_low(30) | Y_high(30) | Z_low(30) | Z_high(30) |
| FIFO（31） | X_low(31) | X_high(31) | Y_low(31) | Y_high(31) | Z_low(31) | Z_high(31) |

**表 36. FIFO 溢出示例（存储第 33 个采样集同时丢弃第 1 个采样）**

| 输出寄存器 | 0x28h | 0x29h | 0x2Ah | 0x2Bh | 0x2Ch | 0x2Dh |
|---|---|---|---|---|---|---|
| | X_low(1) | X_high(1) | Y_low(1) | Y_high(1) | Z_low(1) | Z_high(1) |
| FIFO 索引 | FIFO 样本集合 | | | | | |
| FIFO（0） | X_low(1) | X_high(1) | Y_low(1) | Y_high(1) | Z_low(1) | Z_high(1) |
| FIFO（1） | X_low(2) | X_high(2) | Y_low(2) | Y_high(2) | Z_low(2) | Z_high(2) |
| FIFO（2） | X_low(3) | X_high(3) | Y_low(3) | Y_high(3) | Z_low(3) | Z_high(3) |
| FIFO（3） | X_low(4) | X_high(4) | Y_low(4) | Y_high(4) | Z_low(4) | Z_high(4) |
| ... | ... | ... | ... | ... | ... | ... |
| FIFO（30） | X_low(31) | X_high(31) | Y_low(31) | Y_high(31) | Z_low(31) | Z_high(31) |
| FIFO（31） | X_low(32) | X_high(32) | Y_low(32) | Y_high(32) | Z_low(32) | Z_high(32) |

表 35. FIFO 缓冲区填满示例（存储第 32 个采样集）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，LIS2DH12 输出寄存器（28h 到 2Dh）始终会包含最早的 FIFO 样本集合。

## 9.2 FIFO 寄存器

FIFO 缓冲区由三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，第三个寄存器会提供关于缓冲区状态的信息。

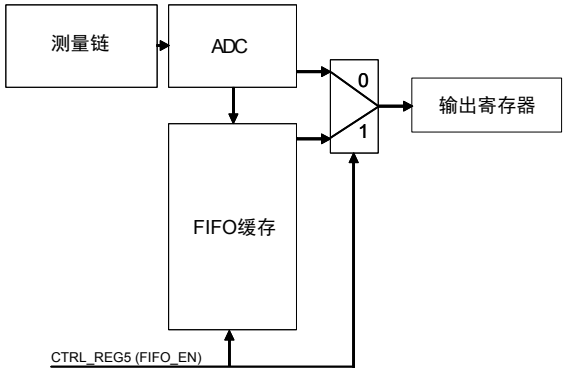
### 9.2.1 CTRL_REG5 (0x24)

CTRL_REG5 中的 FIFO_EN 位必须设为 1 才能使能内部的先进先出缓冲区；如果该位置 1，加速度计输出寄存器（28h 到 2Dh）不会包含当前加速度计值，但始终会包含 FIFO 中存储的最早值。

**表 37. CTRL_REG5 中的 FIFO 使能位**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | FIFO_EN | X | X | X | X | X | X |

**图 24. FIFO_EN 连接框图**




### 9.2.2 FIFO_CTRL_REG (0x2E)

该寄存器专用于 FIFO 模式选择和水印配置。

**表 38. FIFO_CTRL_REG**

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| FM1 | FM0 | TR | FTH4 | FTH3 | FTH2 | FTH1 | FTH0 |

FM[1:0]位定义 FIFO 缓冲区特性的选择：

1. FM[1:0] = (0,0)：Bypass 模式
2. FM[1:0] = (0,1)：FIFO 模式
3. FM[1:0] = (1,0)：Stream 模式
4. FM[1:0] = (1,1)：Stream-to-FIFO 模式

用于激活 Stream-to-FIFO 模式的触发与已选 NT1_SRC 寄存器 IA 位的值相关，不取决于中断引脚值和极性。如果已选中断未驱动到中断引脚，也会生成触发。

FTH[4:0]位定义水印等级；如果 FIFO 内容超过该值，FIFO 源寄存器中的 WTM 位会置“1”。

### 9.2.3 FIFO_SRC_REG (0x2F)

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 39. FIFO_SRC_REG

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| WTM | OVRN_ FIFO | 空 | FSS4 | FSS3 | FSS2 | FSS1 | FSS0 |

- 如果 FIFO 内容超过水印等级，WTM 位会置 1。
- 如果 FIFO 缓冲区已满，OVRN_FIFO 位会置 1，这意味着 FIFO 缓冲区包含 32 个未读样本。下一 ODR 时，新样本集合会替换最早的 FIFO 值。第一个样本集合已被读取时，OVRN_FIFO 位会复位。
- 当所有 FIFO 样本已被读取并且 FIFO 为空时，EMPTY 标志会置 1。
- FSS[4:0]字段始终包含 FIFO 缓冲区中存储的当前未读样本数。FIFO 使能后，该值会以 ODR 频率增加，直至缓冲区已满，随后，每次从 FIFO 读取一个样本集合，该值都会减小。

寄存器内容会与 FIFO 写操作和读操作同步更新。

表 40. FIFO_SRC_REG 特性（假定 FTH[4:0] = 15）

| WTM | OVRN_FIFO | 空 | FSS[4:1] | 未读 FIFO 样本 | 时序 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 00000 | 0 | t0 |
| 0 | 0 | 0 | 00001 | 1 | t0 + 1/ODR |
| 0 | 0 | 0 | 00010 | 2 | t0 + 2/ODR |
| ... | ... | ... | ... | ... | ... |
| 0 | 0 | 0 | 01111 | 15 | t0 + 15/ODR |
| 1 | 0 | 0 | 10000 | 16 | t0 + 16/ODR |
| ... | ... | ... | ... | ... | ... |
| 1 | 0 | 0 | 11110 | 30 | t0 + 30/ODR |
| 1 | 0 | 0 | 11111 | 31 | t0 + 31/ODR |
| 1 | 1 | 0 | 11111 | 32 | t0 + 32/ODR |

可通过配置 CTRL_REG3 使能水印标志和 FIFO 上溢事件，以便在 INT1 引脚上生成专用中断。

表 41. CTRL_REG3 (0x22)

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| X | X | X | X | X | I1_WTM | I1_OVERRUN | X |

- I1_WTM 位会驱动 INT1 引脚上的水印标志(WTM)。
- I1_OVERRUN 位会驱动 INT1 引脚上的上溢事件(OVRN)。

如果两个位均置“1”，INT1 引脚状态为两个信号的逻辑或组合。

## 9.3 FIFO 模式

LIS2DH12 FIFO 缓冲区可配置为在四种不同的模式下工作，可通过 FIFO_CTRL_REG 中的 FM[1:0]字段选择工作模式。提供的配置可确保实现高度灵活性，并可增加应用开发中可使用的功能数。

Bypass、FIFO、Stream 和 Stream-to-FIFO 模式在下面几段中进行了介绍。

### 9.3.1 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。

请按照以下步骤配置 Bypass 模式：

1. 将 CTRL5 寄存器（0x24）中的 FIFO_En 位置“1”可使能 FIFO。执行完此操作后，FIFO 缓冲区会使能，但不会采集数据，数据寄存器冻结为上次加载的样本集合。
2. 将 FIFO_CTRL_REG（0x2E）中的 FM[1:0]字段设为“00”可激活 Bypass 模式。如果使能该模式，FIFO_SRC_REG（0x2F）会强制设为等于 0x20。

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。

### 9.3.2 FIFO 模式

在 FIFO 模式下，缓冲区会继续填入数据，直至填满为止（存储 32 个样本集合），随后，缓冲区会停止采集数据，FIFO 内容保持不变，直至选择了另一模式。

请按照以下步骤配置 FIFO 模式：

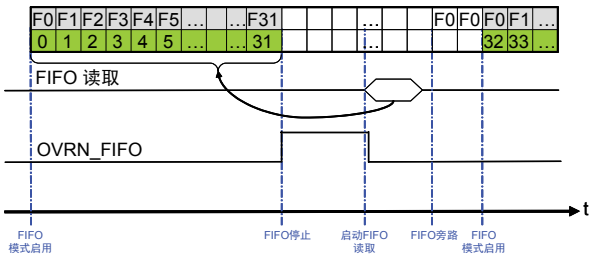
1. 将 CTRL5 寄存器（0x24）中的 FIFO_En 位置“1”可使能 FIFO。执行完此操作后，FIFO 缓冲区会使能，但不会采集数据，数据寄存器冻结为上次加载的样本集合。
2. 将 FIFO_CTRL_REG（0x2E）中的 FM[1:0]字段设为“01”可激活 FIFO 模式。

选择该模式后，FIFO 会开始进行数据采集，FIFO_SRC_REG（0x2F）也会根据存储的样本数发生变化。此过程结束时，如果选择了 CTRL_REG5 中的 I1_OVERRUN 位，FIFO_SRC_REG 会设为 0xDF，OVRN 标志会生成中断。如果 OVRN_FIFO 置为“1”，可重新获取数据，获取数据时会从输出寄存器读取 32 个样本集合，如果应用要求的样本数较少，还可以根据 WTM 标志（而不是 OVRN_FIFO）重新获取数据。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。

FIFO 模式建议如下：

1. 将 FIFO_EN 置 1：使能 FIFO
2. 将 FM[1:0]设为(0,1)：使能 FIFO 模式
3. 等待 OVRN_FIFO 或 WTM 中断；
4. 从加速度计输出寄存器读取数据
5. 将 FM[1:0]设为(0,0)：使能 Bypass 模式
6. 重复第 2 步及后续步骤

**图 25. FIFO 模式特性**




如果使能了 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 OVRN_FIFO 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。读取程序包括一组 32 个 6 字节样本（共 192 字节），会从 FIFO 中存储的最早的样本（F0）开始获取数据。第一个样本集合已被读取时，OVRN_FIFO 位会复位。旁路模式设置会复位 FIFO 并允许用户再次使能 FIFO 模式。

### 9.3.3 Stream 模式

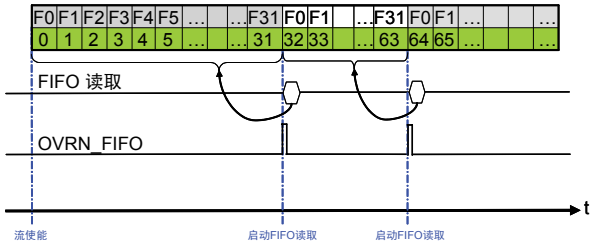
在 Stream 模式下，FIFO 会继续填入数据，如果缓冲区已满，FIFO 索引会从头开始使用，较早的数据会被当前数据替代。最早先的数据继续被覆盖，直至读取操作释放了 FIFO 插槽。要实现插槽释放速度快于新数据产生速度，主处理器读取速度很重要。FM[1:0] Bypass 配置用于停止该模式。

请按照以下步骤配置 FIFO 模式：

1. 将 CTRL5 寄存器（0x24）中的 FIFO_En 位置“1”可使能 FIFO。执行完此操作后，FIFO 缓冲区会使能，但不会采集数据，数据寄存器冻结为上次加载的样本集合。
2. 将 FIFO_CTRL_REG（0x2E）中的 FM[1:0]字段设为“10”可激活 Stream 模式。

如上所述，对于 FIFO 模式，如果 OVRN_FIFO 置为“1”，可读取数据，获取数据时会从输出寄存器读取 32 个样本集合。如果应用要求的样本数较少，还可以根据 WTM 标志重新获取数据。

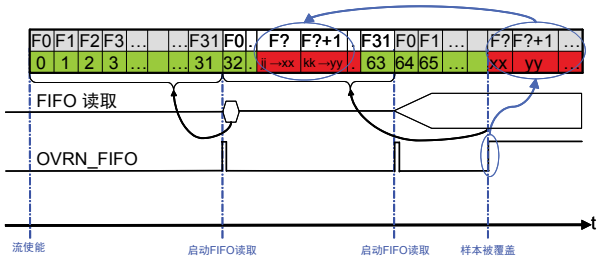
**图 26. Stream 模式快速读取特性**




在 Stream 模式下，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）。缓冲区填满后，OVRN_FIFO 标志会变为高电平，建议的解决方法是以快于 1*ODR 的速度读取所有 FIFO 样本（192 个字节），以便释放 FIFO 位置供新的加速度样本使用。这样可避免数据丢失并减少主机处理器干预，从而提高系统效率。如果读取过程的速度不够快，可观察到三种不同的情况：

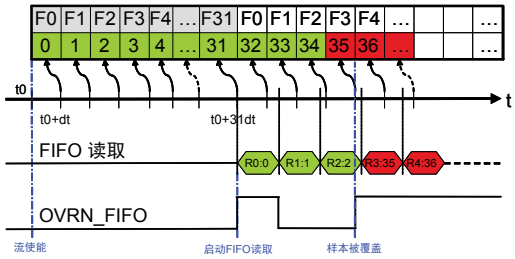
1. FIFO 样本集合（6 字节）的读取速度快于 1*ODR：由于新数据生成前已释放了 FIFO 位置，因此可正确地重新获取数据。
2. FIFO 样本集合（6 字节）的读取速度与 1*ODR 同步：由于新数据生成前已释放了 FIFO 位置，因此可正确地重新获取数据，但没有用到 FIFO 的优势。这种情况相当于在发生数据就绪中断时读取数据，与标准的加速度计读取相比，不会限制主机处理器的干预。
3. FIFO 样本集合（6 字节）的读取速度比 1*ODR 慢：在这种情况下，一些数据会丢失，因为数据恢复的速度不够快，无法为新的加速度数据释放 FIFO 位置。图 27. Stream 模式慢速读取特性.正确恢复的样本数与当前 ODR 与 FIFO 样本集合读取速率之差相关。

**图 27. Stream 模式慢速读取特性**




在图 27. Stream 模式慢速读取特性中，由于读取速度慢，不会重新获取来自“jj”的数据，因为这些数据会被系统生成的新加速度计样本所代替。

图 28. **Stream 模式慢速读取放大图**




使能 Stream 模式后，FIFO 位置会在每个 ODR 时间帧结束时填入数据。OVRN 标志置“1”后，必须立即开始读取过程，会在读取开始时从 FIFO 重新获取数据。读取命令发送到器件后，输出寄存器内容会移动到 SPI/I$^2$C 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。如果读取速度比 1*ODR 慢，新样本插入到寻址位置后，可从 FIFO 重新获取一些数据。在图 28. Stream 模式慢速读取放大图中，F3 索引刷新后会开始执行第四条读取命令，此命令会中断数据读取。OVRN 标志告知用户该事件已发生。本例中读取了三个正确样本，正确恢复的样本数取决于 ODR 和 FIFO 样本集合读取时间帧之差。

### 9.3.4 sStream-to-FIFO 模式

此模式是上述 Stream 模式与 FIFO 模式的结合。在 Stream-to-FIFO 模式下，FIFO 缓冲区会在 Stream 模式下开始工作，并会在发生选定的中断后切换为 FIFO 模式。

请按照以下步骤配置 Stream-to-FIFO 模式：

1. 使用寄存器 INT1_CFG (0x30)配置所需中断发生器。
2. 根据配置的中断发生器配置 FIFO_CTRL_REG（0x2E）寄存器中的 TR 位：将 TR 设为“0”可选择中断 1，将 TR 设为“1”可选择中断 2。
3. 将 CTRL5 寄存器（0x24）中的 FIFO_En 位置“1”可使能 FIFO。执行完此操作后，FIFO 缓冲区会使能，但不会采集数据，数据寄存器冻结为上次加载的样本集合。
4. 将 FIFO_CTRL_REG（0x2E）中的 FM[1:0]字段设为“11”可激活 Stream-to-FIFO 模式。

中断触发与 INT1_SRC 寄存器中的 IA 相关联，即使中断信号未驱动到中断焊盘，也会生成中断触发。如果 IA 和 OVRN_FIFO 位均置 1，则会执行模式转换。Stream-to-FIFO 模式易受触发电平影响，而不是是触发边沿的影响；这意味着如果 stream-to-FIFO 处于 FIFO 模式，并且中断条件消失，FIFO 缓冲区会因 IA 位变为零而恢复为 Stream 模式。建议锁存用作 FIFO 寄存器的中断信号，以免中断信号丢失。如果选定的中断被锁存，则必须读取寄存器 INT1_SRC 将 IA 位清零；读取后，IA 位会取 2*ODR 的值变低。

在 Stream 模式下，FIFO 缓冲区会继续填入数据，当缓冲区已满时，OVRN_FIFO 位会置 1，后续样本会覆盖原样本。发生触发时，可观察到两种不同情况：

1. 如果 FIFO 缓冲区已满(OVRN_FIFO = “1”)，则会在触发后收到第一个样本时停止采集数据。FIFO 内容由触发事件前的 30 个样本、生成了中断的样本以及触发后的一个样本构成。
2. 如果 FIFO 未满（初始瞬态），则会继续填入数据，直至填满为止(OVRN_FIFO = “1”)，之后，如果触发条件仍存在，FIFO 会停止采集数据。

图 29. **Stream-to-FIFO 模式：中断未锁存**

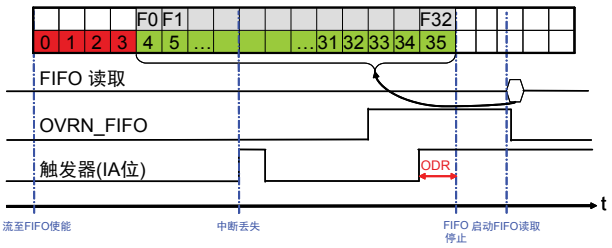
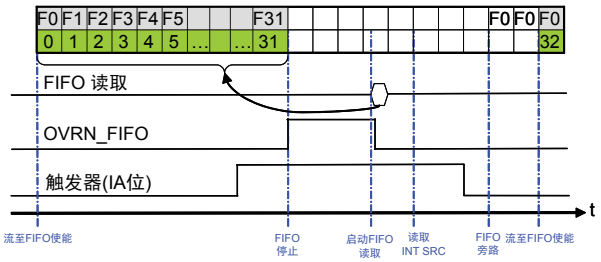


图 30. **Stream-to-FIFO 模式：中断锁存**



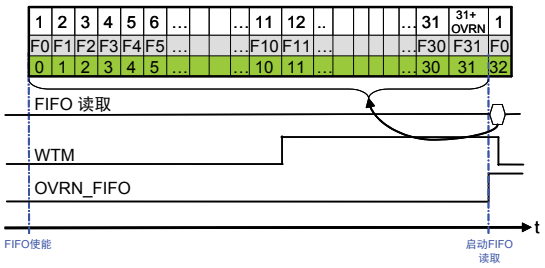

可使用 Stream-to-FIFO 来分析生成中断的样本历史；标准操作是在 FIFO 模式已触发、FIFO 缓冲区已满并停止时读取 FIFO 内容。

## 9.4 水印

水印是可用于生成特定中断的可配置标志，可用于确定 FIFO 缓冲区何时包含的样本数至少为定义为水印级别的数目。用户可使用 FIFO_CTRL_REG 中的 FTH[4:0]字段选择所需级别（范围为 0 到 31），而 FIFO_SRC_REG FSS[4:0]始终包含存储在 FIFO 中的样本数。

如果 FSS[4:0]大于 FTH[4:0]，FIFO_SRC_REG 中的 WTM 位会置 1，相反，如果 FSS[4:0]字段小于 FTH[4:0]，WTM 会置 0。FSS[4:0]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，FSS[4:0]会减小一步。

图 31. **水印特性 - FTH[4:0] = 10**




在图 31. 水印特性 - FTH[4:0] = 10 中，第一行指示 FSS[4:0]值，第二行指示相对 FIFO 位置，最后一行显示增加的 FIFO 数据。假定 FTH[4:0] = 10，当第 11 个 FIFO 位置(F10)被填入数据时，WTM 标志会从“0”变为“1”。
图 32. FIFO 读取图 - FTH[4:0] = 10 显示了当 FIFO 内容小于 FTH[4:0]时，WTM 标志变为低电平，这意味着第 9 个未读样本集合仍在 FIFO 中。

可将 CTRL_REG3 中的 I1_WTM 位置为高电平，以此使能水印标志(WTM)，在 INT1 引脚上生成专用中断。

## 9.5 从 FIFO 读取数据

如果 FIFO 已使能，并且所处模式不是 Bypass 模式，读取输出寄存器（28h 到 2Dh）会返回最早的 FIFO 样本集合。

读取输出寄存器时，其内容会移至 SPI/I$^2$C 输出缓冲区。FIFO 位置最好上移一位，以便为接收的新样本释放空间，并使输出寄存器能够载入 FIFO 缓冲区中存储的当前最早的值。

整个 FIFO 内容是通过对加速度计输出寄存器执行 32 次读取操作读取的，FIFO 缓冲区有新样本集合之前，所有其他的读取操作都会返回相同的最后值。

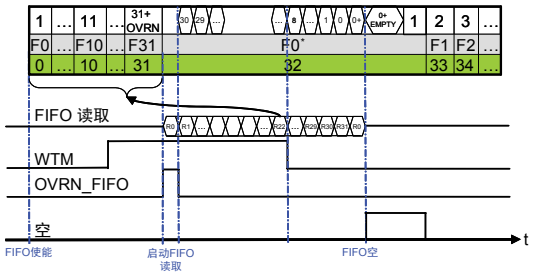
为了提高应用的灵活性，可使用每种读取字节组合从 FIFO 重新获取数据（例如：196 次单字节读取，32 次 6 字节读取，1 次 196 字节的多字节读取等）。

建议以快于 1*ODR 的速度通过 196 字节多字节读取的方式来读取所有 FIFO 位置（6 个输出寄存器乘以 32 个存储位置）。为了最大限度地减少主器件与从器件之间的通信，器件会自动更新读取地址；到达寄存器 0x2D 时，会回滚到 0x28。

为了避免数据丢失，必须按照可用的串行通信速率选择正确的 ODR。如果使用的是标准 I$^2$C 模式（最大速率 100 kHz），单次样本集合读取会用时 830 µs，而整体 FIFO 下载约用时 17.57 ms。I$^2$C 速度低于 SPI，大概需要 29 个时钟脉冲才能开始通信（开始、从站地址、器件地址+写入地址、重新开始、器件地址+读取），另外还需要 9 个时

钟脉冲才能读取每个字节。如果遵循该建议，完整 FIFO 读取的执行速度会快于 1*ODR，这意味着如果使用标准 I²C，可选 ODR 必须低于 57 Hz。如果使用快速 I²C 模式（最大速率 400 kHz），可选 ODR 必须低于 228 Hz。

**图 32. FIFO 读取图 - FTH[4:0] = 10**




在图 32. FIFO 读取图 - FTH[4:0] = 10 中，“Rx”表示 6 字节读取操作，“F0*”代表单个 ODR 时隙（为便于阅读进行了放大）。

*注意：为了将 EMPTY 位置为“1”，必须读取并丢弃一个额外样本（最早一个重复样本）。*

# 10 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，器件能够自动将加速度计采样率降低至 10 Hz 低功耗，当检测到唤醒中断事件时增加加速度计 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全性能模式，反之亦然，因此可以保证节能和灵活性。

通过将非零唤醒阈值写入寄存器 ACT_THS 使能活动/不活动识别功能。返回睡眠时间也可进行配置，作用于寄存器 ACT_DUR。

图 33. 活动/不活动识别

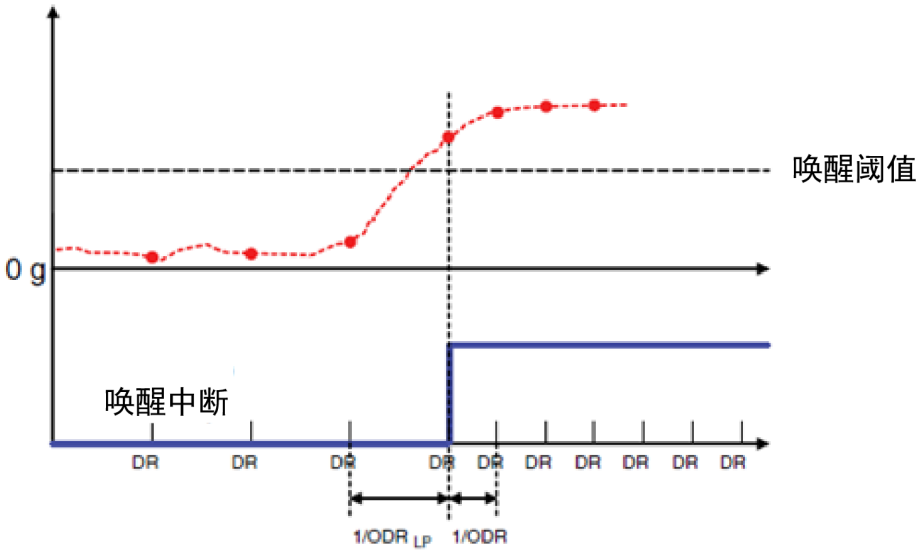
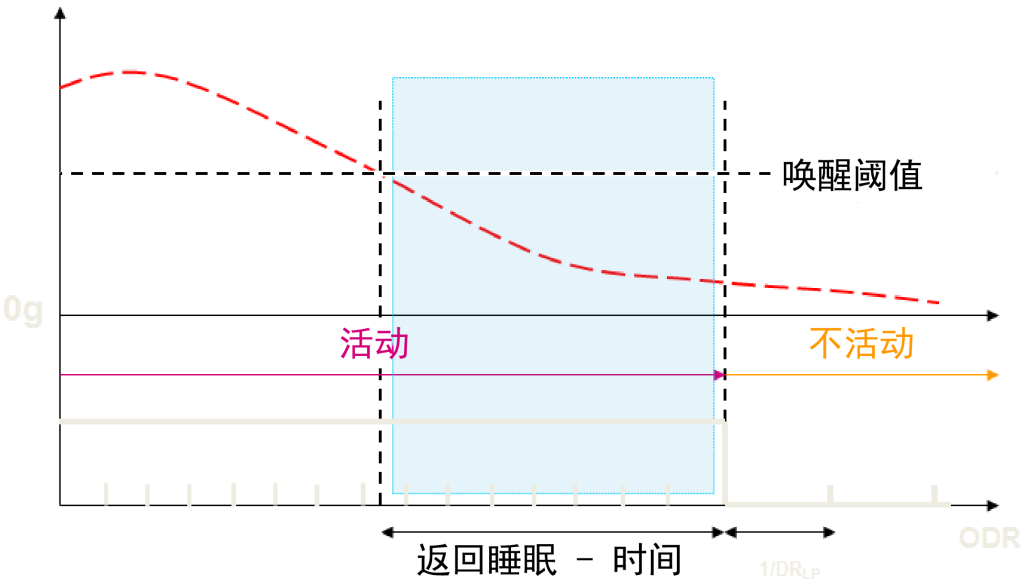



图 34. 活动/不活动持续时间



用户还可以通过将寄存器 I2_ACT 的 I2_ACT 位置为 1，将系统状态（活动/不活动）连接到 INT2 引脚。使用此功能时，INT2 在系统处于不活动状态（ODR 为 10 Hz）时为高电平，并在系统处于活动状态（用户 ODR）时变为低电平。INT_POLARITY 位控制活动/不活动信号的极性。

# 11 温度传感器

为了使能内部温度传感器，寄存器 TEMP_CFG_REG（1Fh）中的 TEMP_EN[1:0]位和寄存器 CTRL_REG4（23h）中的 BDU 位必须置位。

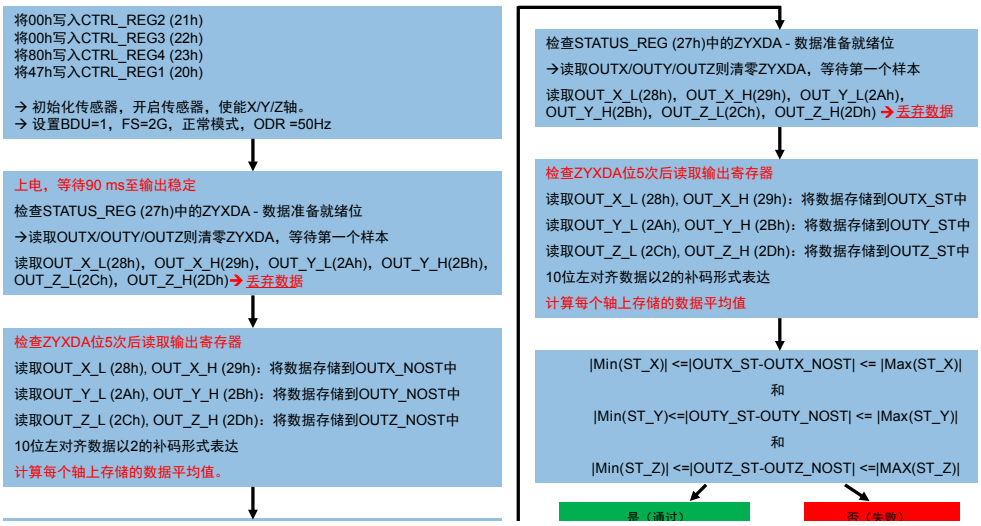
温度作为二进制补码数据（左对齐）保存在 OUT_TEMP_L（0Ch）和 OUT_TEMP_H（0Dh）中。

如果 CTRL_REG1（20h）中的 LPen（位 3）清零，则温度数据格式可以是 10 位（高分辨率/正常模式），否则，在低功耗模式下，ADC 分辨率为 8 位。请参考 LIS2DH12 数据表了解转换系数。

# 12 加速度计自检步骤

自检允许用户检查传感器功能而无需移动传感器。当自检使能时，传感器上会施加一个致动力，模拟一定的数据加速度。这种情况下，传感器输出会在其 DC 电平上表现出变化，该电平通过器件灵敏度关联到所选满量程。当自检功能激活时，器件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。如果输出信号在给定范围（数据表中提供了最小值和最大值）内变化，则传感器正常工作，接口芯片的参数在定义范围内。图 35. 自检步骤中描述了自检步骤。

图 35. 自检步骤



# 版本历史

表 42. 版本历史

| 日期 | 版本 | 变更 |
|---|---|---|
| 2017 年 2 月 10 日 | 1 | 初始版本 |
| 2017 年 6 月 08 | 2 | 更新了 Section 3.4 高分辨率模式和 Section 8 单击和双击识别<br>更新了表 9. 输出数据寄存器内容 vs. 加速度（FS = ±2 *g*，高分辨率模式）和 表 22. 真值表<br>增加了 Section 12 加速度计自检步骤 |