DT0098
Design tip

Setting up wake-up recognition and absolute/relative references with ST's MEMS accelerometers

By Vladimir JANOUSEK, Zuzana JIRANKOVA and Petr STUKJUNGER

Main components	
LIS2DW12	MEMS digital output motion sensor: high-performance ultra-low-power 3-axis "femto" accelerometer
LIS2DH12	MEMS digital output motion sensor: ultra-low-power high-performance 3-axis "femto" accelerometer

Purpose and benefits

This design tip explains how to enable and personalize the wake-up recognition feature of MEMS accelerometers from STMicroelectronics.

First we explain this embedded feature, what it does and how it can be utilized in applications. Then we discuss the impact of its parameters on wake-up recognition results. Finally we show using the two most frequently used ST accelerometers, LIS2DW12 and LIS2DH12, exact settings and example source codes for implementing this feature in applications.

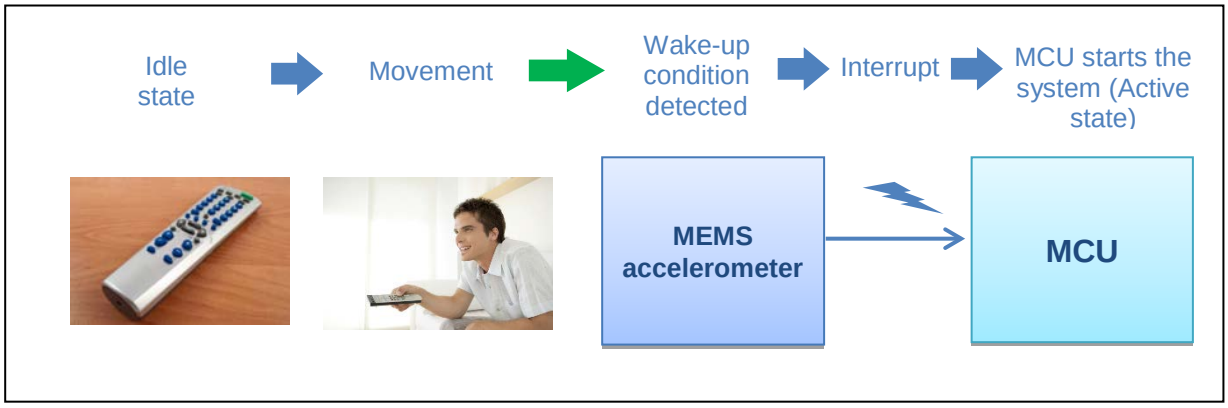
Description

Wake-up recognition is a feature of an accelerometer allowing autonomous detection of acceleration shocks or movements from a stable position, i.e. if someone has moved the device or if the device was moved/rotated from a user predefined position. The wake-up feature is very useful for alarm systems or for smart power management of a battery-operated application having typically two states:

a) **Idle state** where the application is not in use and the majority of the application circuits are switched off beside the accelerometer (e.g. TV remote control is laying on a table, see Figure 1)

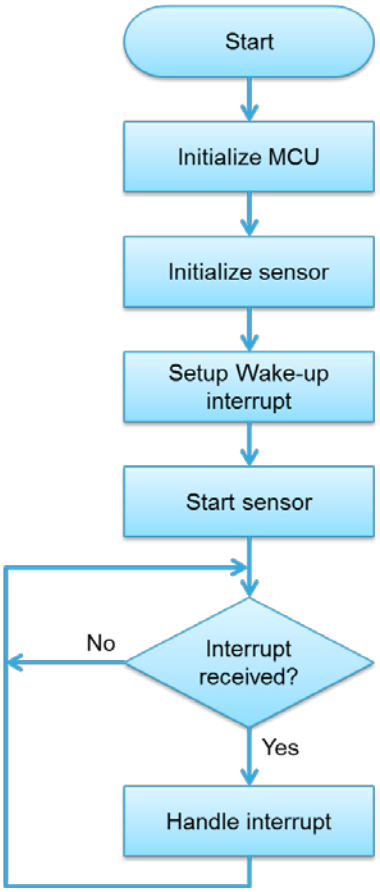
b) **Active state** – the accelerometer wake-up feature helps to turn on the entire application, usually waking up the application MCU (e.g. TV remote control is picked up from the table and used, see Figure 1)

Figure 1. Wake-up event recognition flow



The wake-up can be personalized by two key parameters – **threshold** and **duration**. With **threshold**, the application can set the value which at least one of the axes has to exceed in order to generate an interrupt. The **duration** is the number of consecutive samples for which the measured acceleration exceeds the threshold, see Figure 2.

Flowchart



Wake-up in the LIS2DW12 and LIS2DH12

What was described so far is valid for both the LIS2DW12 and LIS2DH12. However there are some significant differences between the two sensors in functionality, see Table 1, and in the naming convention of the registers when using the wake-up recognition feature.

Table 1. Wake-up functionality options in the LIS2DW12 and LIS2DH12

Wake-up options	LIS2DW12	LIS2DH12
Using the HP filter	X	X
Relative reference mode	X	
Absolute reference mode	X	
HP filter bypassed		X

Usually when the wake-up feature is considered for an application, the key requirement is also to run this function with minimal power consumption of the sensor itself, since the sensor is the only device running in the idle state of the application. The power consumption depends on the sensor output data rate (ODR) and the sensor operating mode selected, see datasheets for more information. Table 2 shows the LIS2DW12 and LIS2DH12 sensor set-up for minimal power consumption, the wake-up recognition using the HP filter feature activated and the interrupt generated on the INT1 pin. Of course it is possible to use higher ODRs and higher performance modes, this depends on the requirements of the application.

Table 2. Example showing register set-up for wake-up recognition

	LIS2DW12	LIS2DH12
Register set-up	CTRL1 = 0x10 CTRL4_INT1_PAD_CTRL = 0x20 WAKE_UP_THS = 0x02 WAKE_UP_DURATION = 0x00 CTRL7 = 0x20	CTRL_REG1 = 0x1F CTRL_REG2 = 0x01 CTRL_REG3 = 0x40 INT1_CFG = 0x2A INT1_THS = 0x10 INT1_DURATION = 0x00
Power consumption [uA]	0.38 (Vdd = 1.8 V)	2 (Vdd = 2.5 V)

✎ *For the LIS2DH12 it is recommended to read the REFERENCE (26h) register when a wake-up interrupt is generated and captured by the MCU. It will reset*

the internal HP filter and will allow the LIS2DH12 to detect any potentially close (timewise) wake-up event.

✎ *In order to achieve minimal power consumption, it is also important to pay attention to the sensor's embedded pull-ups. A common mistake is that the SDO/SA0 pin embedding a pull-up resistor is grounded in the application. It is recommended to connect the pin to Vdd or to leave the pin unconnected. In the LIS2DH12 it is possible to disable the pull-up by changing CTRL_REG0, bit SDO_PU_DISC. In the LIS2DW12 pay attention to leave the SDO/SA0 unconnected or connect it to Vcc.*

Setting up wake-up recognition using the HP filter in the LIS2DW12

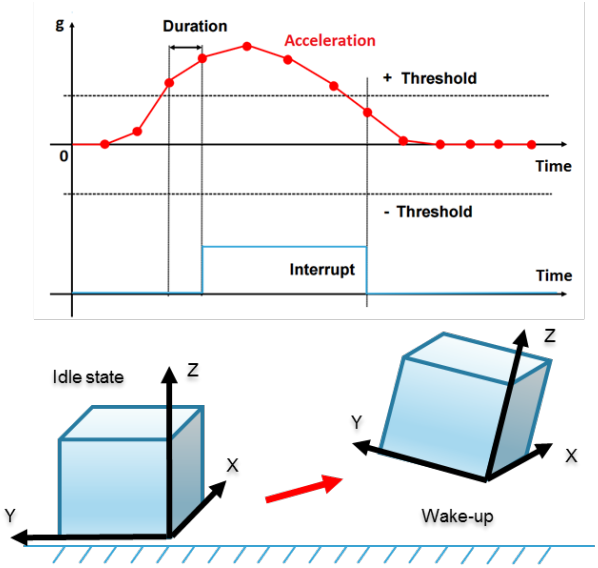
Wake-up recognition is working well regardless of the initial 3D orientation of the sensor in the environment, because the Earth's gravity component of the acceleration is filtered out and only the dynamic motion changes in any direction are detected if they exceed the threshold.

To enable wake-up recognition using the HP filter, you need to:

- Initialize the MCU
- Set bit **INT1_WU** in **CTRL4_INT1_PAD_CTRL** register (23h)
- Set desired threshold to bits **WK_THS[5:0]** in **WAKE_UP_THS** register (34h)
- Set desired duration to bits **WAKE_DUR[1:0]** in **WAKE_UP_DUR** register (35h)
- Set sensor **ODR to 100 Hz** (recommended) using **ODR[3:0]** bits and operating mode to Low-power mode 1 using bits **MODE[1:0]** and **LP_MODE[1:0]** in **CTRL1** register (20h)
- Set bit **INTERRUPTS_ENABLE** in **CTRL7** register (3Fh)

✎ *Set-up of the ODR depends on the requirements of the application. The lower the ODR, the lower the power consumption, but on the other hand, the sensitivity to detect the wake-up event is worse (the time between two samples is longer).*

Figure 2: Wake-up event recognition using the HP filter



The measured acceleration is 0 *g* in the Idle state (the HP filter eliminates the acceleration DC component). Movement generates an acceleration peak rising above 0 *g* on all the corresponding sensor axes.

Pseudocode for wake-up recognition using the HP filter in the LIS2DW12

```
void LIS2DW12_INT1_handler(void)
{
  print("Wakeup detected\r\n");
  /* … */
}
int main(void)
{
  init_MCU();             /* Initialize MCU clock and pins */
  print("Starting program\r\n");

  /* Initialization of sensor */
  write_reg(0x21, 0x08);   /* CTRL2(21h): Enable BDU */
  write_reg(0x25, 0x00);   /* CTRL6(25h): Set Full-scale to +/-2g */

  /* Wake-up recognition enable */
  write_reg(0x23, 0x20);   /* CTRL4_INT1_PAD_CTRL(23h): Enable Wake-up interrupt */
  write_reg(0x34, 0x01);   /* WAKE_UP_THS(34h): Threshold set */
  write_reg(0x35, 0x00);   /* WAKE_UP_DUR(35h): Duration set */

  /* Start sensor */
  write_reg(0x20, 0x50);   /* CTRL1(20h): Set ODR 100Hz,Low-Power mode 1(12 bit) */
  delay(10);               /* Settling time ( 1 sample, i.e. 1/ODR ) */
```

```
  write_reg(0x3f, 0x20);   /* Enable interrupts */

  while (1)
  {
   /* … */
  }
}
```

Settings to enable the absolute reference wake-up in the LIS2DW12 in addition to wake-up recognition using the HP filter in the LIS2DW12

- Set the reference position by writing axis values to **X_OFS_USR**, **Y_OFS_USR** and **Z_OFS_USR** registers (3Ch, 3Dh, 3Eh)
- Set bits **USR_OFF_ON_WU** and **USR_OFF_W** in **CTRL7** register (3Fh)

✎ *The USR_OFF_W bit defines the weight of the user reference bytes specified by the X(Y,Z)_OFS_USR registers. For absolute reference interrupt set-up it is recommended to set this bit, enabling 15.6 mg/LSB weight. This allows setting the X(Y,Z)_OFS_USR bits in the range ±2 g (256 * 15.6 mg = 4 g).*

In the absolute reference mode, the user can define the reference position in which no interrupt is generated by writing to the X_OFS_USR, Y_OFS_USR and Z_OFS_USR registers, see Figure 3.

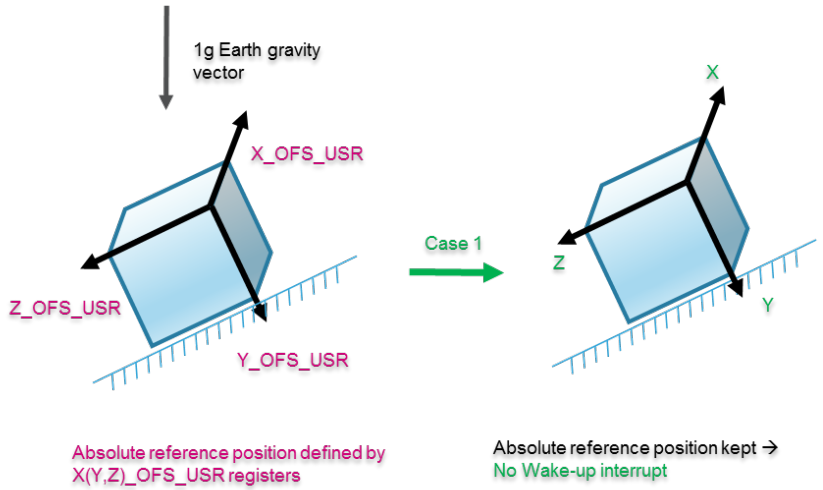
Examples:

X_OFS_USR = 0x00, Y_OFS_USR = 0x00, Z_OFS_USR = 0x40
(X = 0 m*g*, Y = 0 m*g*, Z = 1000 m*g*)

X_OFS_USR = 0x00, Y_OFS_USR = 0x40, Z_OFS_USR = 0x00
(X = 0 m*g*, Y = 1000 *g*, Z = 0 m*g*)

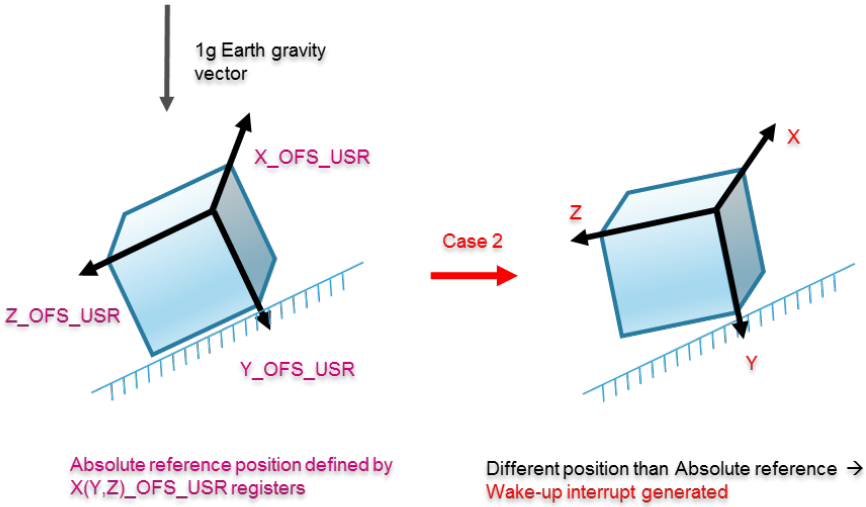
After the sensor setup, if the measured acceleration exceeds the wake-up threshold defined by the WK_THS[5:0] bits for a duration longer than defined by the WAKE_DUR[1:0] bits from the predefined absolute reference position, an interrupt is generated, see Figure 4. If you move/rotate back the system to the original reference position stored in the X_OFS_USR, Y_OFS_USR and Z_OFS_USR registers, the interrupt is deactivated.

Figure 3: Absolute reference wake-up in the LIS2DW12.



If the device is kept in the reference position (orientation) defined by the X(Y,Z) _OFS_USR registers, then no interrupt is generated.

Figure 4: Absolute reference wake-up in the LIS2DW12



If the device is moved (rotated) from the reference position defined by the X(Y,Z) _OFS_USR registers exceeding the WAKE_UP_THS threshold and longer than the WAKE_UP_DUR time, then an interrupt is generated.

Pseudocode for absolute reference wake-up in the LIS2DW12

```
void LIS2DW12_INT1_handler(void)
{
  print("Wakeup detected\r\n");
  …
}
int main(void)
```

```
{
  init_MCU();              /* Initialize MCU clock and pins */
  print("Starting program\r\n");

  /* Initialization of sensor */
  write_reg(0x21, 0x08);   /* CTRL2(21h): Enable BDU
  write_reg(0x25, 0x00);   /* CTRL6(25h): Set Full-scale to +/-2g

  /* Wake-up recognition enable */
  write_reg(0x23, 0x20);   /* CTRL4_INT1_PAD_CTRL(23h): Enable Wake-up interrupt */
  write_reg(0x34, 0x01);   /* WAKE_UP_THS(34h): Threshold set */
  write_reg(0x35, 0x00);   /* WAKE_UP_DUR(35h): Duration set */

  /* Absolute reference set */
  write_reg(0x3c, 0x00);   /* X_OFS_USR(3Ch): X axis absolute reference */
  write_reg(0x3d, 0x00);   /* Y_OFS_USR(3Dh): Y axis absolute reference */
  write_reg(0x3e, 0x40);   /* Z_OFS_USR(3Eh): Z axis absolute reference */

  /* Start sensor */
  write_reg(0x20, 0x20);   /* CTRL1(20h): Set ODR 12.5Hz,Low-Power mode 1(12 bit) */
  delay(80);               /* Settling time ( 1 sample, i.e. 1/ODR ) */
  write_reg(0x3f, 0x2C);   /* CTRL7(3Fh): Enable interrupts and absolute reference
*/

  while (1)
  {
    /* … */
  }
}
```

Settings to enable relative reference wake-up in the LIS2DW12 in addition to wake-up recognition using the HP filter in the LIS2DW12

- Rotate the sensor to the position you want to be the reference position
- Set bit **HP_REF_MODE** in **CTRL7** register (3Fh)

✎ *If you want to detect wake-up from a different position, disable the HP_REF_MODE bit, rotate the device in the new position and set the HP_REF_MODE bit again.*

This mode is used when the user wants to save the actual measured acceleration as a reference value without the need to write X_OFS_USR, Y_OFS_USR and Z_OFS_USR registers in comparison to the absolute reference wake-up mode. The reference position is stored by writing the HP_REF_MODE bit in CTRL7 register internally in the sensor. Then if the measured acceleration exceeds the wake-up threshold defined by the WK_THS[5:0] bits and for a duration longer than defined by the WAKE_DUR[1:0] bits, an interrupt is generated. If you move the board back to the original reference position, the interrupt is deactivated. The behavior is otherwise the same as shown in Figure 3 and Figure 4.

Pseudocode for relative reference wake-up in the LIS2DW12

```
void LIS2DW12_INT1_handler(void)
{
  print("Wakeup detected\r\n");
  /* … */
}
int main(void)
{
  init_MCU();             /* Initialize MCU clock and pins */
  print("Starting program\r\n");

  /* Initialization of sensor */
  write_reg(0x21, 0x08);   /* CTRL2(21h): Enable BDU */
  write_reg(0x25, 0x00);   /* CTRL6(25h): Set Full-scale to +/-2g */

  /* Wake-up recognition enable */
  write_reg(0x23, 0x20);   /* CTRL4_INT1_PAD_CTRL(23h): Enable Wake-up interrupt */
  write_reg(0x34, 0x01);   /* WAKE_UP_THS(34h): Threshold set */
  write_reg(0x35, 0x00);   /* WAKE_UP_DUR(35h): Duration set */

  /* Start sensor */
  write_reg(0x20, 0x20);   /* CTRL1(20h): Set ODR 12.5Hz,Low-Power mode 1(12 bit) */
  delay(80);               /* Settling time ( 1 sample, i.e. 1/ODR ) */
  write_reg(0x3f, 0x20);   /* CTRL7(3Fh): Enable interrupts */

  write_reg(0x3f, 0x22);   /* CTRL7(3Fh): Save position */

  while (1)
  {
    /* … */
  }
}
```

Setting up wake-up recognition using the HP filter in the LIS2DH12

Wake-up recognition in the LIS2DH12 is working same way as in the LIS2DW12. It works regardless of the initial 3D orientation of the sensor in the environment, because the Earth's gravity component of the acceleration is filtered out by the HP filter and only the dynamic motion changes in any direction are detected if they exceed the threshold. See Figure 2 for more details, the HP filter inertial wake-up works as in the LIS2DW12.

To enable wake-up recognition using the HP filter in the LIS2DH12 you need to:

- Initialize the MCU
- Set bits **FDS** and **HP_IA1** in **CTRL_REG2** register (21h)

- Set bit **I1_IA1** in **CTRL_REG3** register (22h)
- Set bits **ZHIE**, **YHIE**, **XHIE** in **INT1_CFG** register (30h)
- Set desired threshold to bits **THS [6:0]** in **INT1_THS** register (32h)
- Set desired duration to bits **D [6:0]** in **INT1_DURATION** register (33h)
- Start sensor with **ODR low-power 100 Hz** (recommended) - bits **ODR[3:0]** and **LPen** bit in **CTRL_REG1** register (20h)

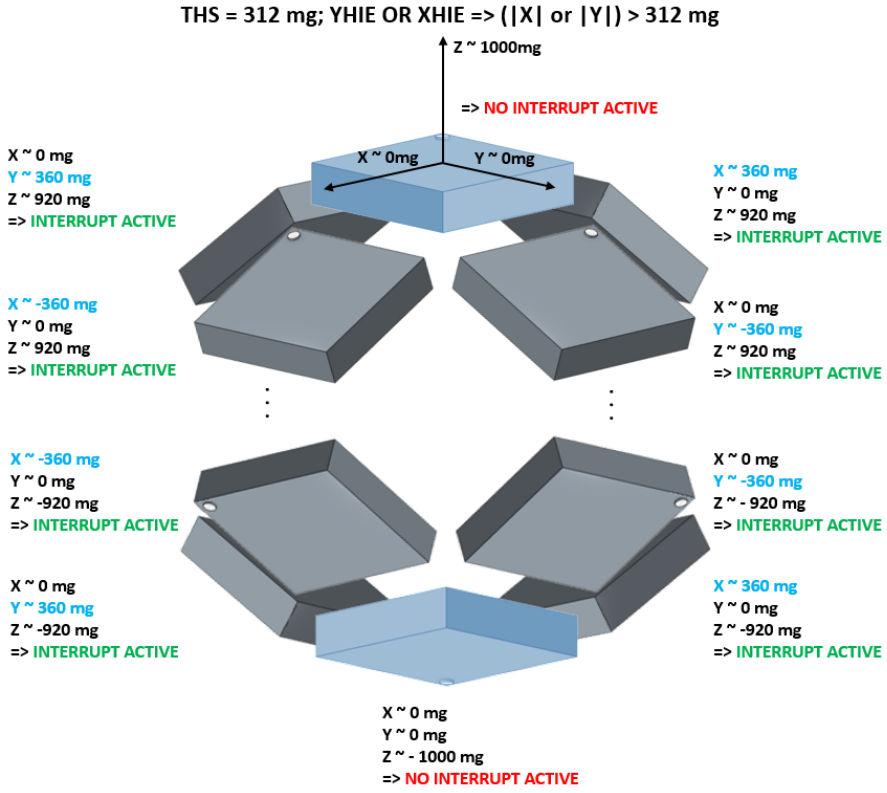
Inertial wake-up recognition using the HP filter in the LIS2DH12

```
void LIS2DH12_INT1_handler(void)
{
  print("Wakeup detected\r\n");
  /* … */
}
int main(void)
{
  init_MCU();               /* Initialize MCU clock and pins */
  print("Starting program\r\n");

  /* Initialization of sensor */
  write_reg(0x21, 0x09);     /* CTRL_REG2(21h): Filtered data and High-pass filter
selection */
  write_reg(0x22, 0x40);     /* CTRL_REG3(22h): IA1 interrupt on INT1 pin */
  write_reg(0x23, 0x00);     /* CTRL_REG4(23h): Set Full-scale to +/-2g */

  /* Wakeup recognition enable */
  write_reg(0x30, 0x2a);     /* INT1_CFG(30h): INT1 Configuration */
  write_reg(0x32, 0x14);     /* INT1_THS(32h): INT1 Threshold set */
  write_reg(0x33, 0x00);     /* INT1_DURATION(33h): INT1 Duration set */

  /* Start sensor */
  write_reg(0x20, 0x5f);     /* CTRL_REG1(20h): Start sensor at ODR 100Hz Low-power
mode
  HAL_Delay(1);              /* Settling time 1ms */

  while (1)
  {
    /* … */
  }
}
```

Bypassing the HP filter for wake-up recognition in the LIS2DH12

The LIS2DH12 allows the logical combination (logical AND or logical OR) of the overthreshold and the underthreshold events for each of the three sensor axis. There are two INT1_CFG, INT2_CFG register descriptions in the LIS2DH12 datasheet showing options to set ZHIE (Z high event), ZLIE (Z low event), YHIE (Y high event), YLIE (Y low event), XHIE (Y high event), XLIE (X low event) events. In theory there are 64 possible combinations for logical OR and same count also for logical AND combinations. The selection to use logical AND or logical OR combination is possible in INT1_CFG, INT2_CFG registers as well, using bits AOI and 6D (AOI=0, 6D=0 is OR combination; AOI=1, 6D=0 is AND combination).

Figure 5: HP filter bypassed for wake-up recognition in the LIS2DH12



This example shows the logical OR combination of YHIE (Y high event), XHIE (X high event). An interrupt is generated only if the measured acceleration on any of the two axes (X or Y) exceeds the defined threshold (INT1_THS).

In the following paragraph we also describe only one of the options, as an example, which is the logical OR combination of the YHIE, XHIE cases, see also Figure 5. In order to bypass the HP filter (wake-up recognition), you need to:

- Initialize the MCU
- Set bit **I1_IA1** in **CTRL_REG3** register (22h)
- Set bits **YHIE**, **XHIE** in **INT1_CFG** register (30h)

- Set desired threshold to bits **THS [6:0]** in **INT1_THS** register (32h)
- Set desired duration to bits **D [6:0]** in **INT1_DURATION** register (33h)
- Start sensor with **ODR low-power 100 Hz** (recommended) - bits **ODR[3:0]** and **LPen** bit in **CTRL_REG1** register (20h)

HP filter bypassed, wake-up pseudocode in the LIS2DH12

```
void LIS2DH12_INT1_handler(void)
{
  print("Wakeup detected\r\n");
  /* … */
}
int main(void)
{
  init_MCU();                /* Initialize MCU clock and pins */
  print("Starting program\r\n");

  /* Initialization of sensor */
  write_reg(0x22, 0x40);     /* CTRL_REG3(22h): IA1 interrupt on INT1 pin */
  write_reg(0x23, 0x00);     /* CTRL_REG4(23h): Set Full-scale to +/-2g */

  /* Wakeup recognition enable */
  write_reg(0x30, 0x0a);     /* INT1_CFG(30h): INT1 Configuration */
  write_reg(0x32, 0x14);     /* INT1_THS(32h): INT1 Threshold set */
  write_reg(0x33, 0x00);     /* INT1_DURATION(33h): INT1 Duration set */

  /* Start sensor */
  write_reg(0x20, 0x5f);     /* CTRL_REG1(20h): Start sensor at ODR 100Hz Low-power
mode */
  HAL_Delay(1);              /* Settling time 1ms */

  while (1)
  {
    /* … */
  }
}
```

Support material

Related design support material
Product evaluation board – X-NUCLEO-IKS01A2, Motion MEMS and environmental sensor expansion board for STM32 Nucleo
Product evaluation board – STEVAL-MKI179V1, LIS2DW12 adapter board for a standard DIL 24 socket
Product evaluation board – STEVAL-MKI151V1, LIS2DH12 3-axis accelerometer adapter board for standard DIL 24 socket, compatible with STEVAL-MKI109V2
Documentation
Datasheet LIS2DW12, High-performance ultra-low-power 3-axis "femto" accelerometer
Datasheet LIS2DH12, High-performance ultra-low-power 3-axis "femto" accelerometer
Application note AN5038, LIS2DW12: MEMS digital output motion sensor ultra-low-power high-performance 3-axis "nano" accelerometer
Application note AN5005, LIS2DH12: MEMS digital output motion sensor ultra-low-power high-performance 3-axis "nano" accelerometer

Revision history

Date	Version	Changes
04-May-2018	1	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

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

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

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

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

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

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