life.augmented

DT0102
Design tip

Setting up single data conversion mode in the LIS2DW12

By Zuzana JIRANKOVA, Petr STUKJUNGER, and Vladimir JANOUSEK

Main components	
LIS2DW12	High-performance ultra-low-power 3-axis "femto" accelerometer

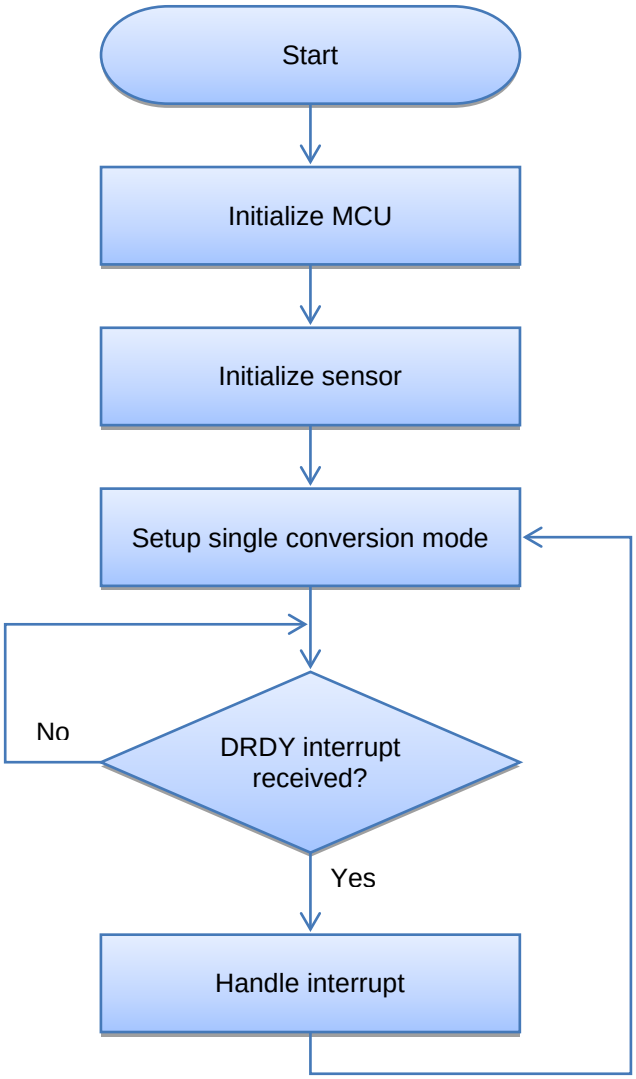
Purpose and benefits

This design tip explains how to enable and personalize the single data conversion feature of MEMS accelerometers from STMicroelectronics with focus on the LIS2DW12. An explanation of how to utilize this feature in order to reduce system power consumption compared to the standard approach with a fixed sensor output data rate is presented. Furthermore we explain the advantage of the single data conversion mode in synchronization with the application running in the MCU with and without the use of the LIS2DW12 embedded FIFO.

Description

The single mode data conversion mode enables reducing power consumption of the accelerometer compared to its normal operation mode (with fixed output data rate, ODR) because the sensor spends the majority of time in power-down mode, preserving only the content of the registers (50 nA). The single data conversion can be also called data on-demand mode because only the application MCUs decide when the data should be measured by the sensor. The advantage of the single data conversion mode is the possibility to synchronize the sensor data conversion / reading to the needs of the application running in the MCU without any minimum duty cycle limit. In this scenario the MCU wakes up the sensor from power-down mode to measure one single set of acceleration data per axis after which the sensor returns automatically to power-down mode. Two approaches are possible and will be described in the following paragraphs. We explain both an I²C/SPI command approach as well as a trigger through a sensor input pin, see the application flowchart. In the last section, utilization of the single mode data conversion with FIFO buffer is also described.

Flowchart



Setting up single data conversion over an I²C/SPI command

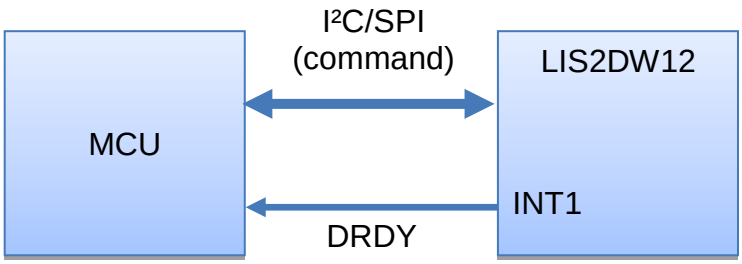
If the SLP_MODE_SEL bit in CTRL3 (22h) is set to 1, the application MCU can start the single data sensor conversion by setting the SLP_MODE_1 in CTRL3 (22h) through the I²C/SPI register write. When set, the sensor goes from power-down mode to data conversion mode. When sensor data are available in the sensor output registers, the SLP_MODE_1 bit is automatically cleared, the sensor goes automatically in power-down mode and becomes ready for another trigger from the MCU.

There are three ways to wait for data availability in the sensor output data registers. The application can either configure the data-ready interrupt - DRDY (as shown in the pseudocode) or the application can continuously poll either the SLP_MODE_1 bit status in register CTRL3 (22h) or the DRDY bit in STATUS (27h) register by an I²C/SPI read. The hardware connections are illustrated in Figure 1.

To enable single data conversion over an I²C/SPI command, the application has to:

- Initialize the MCU
- Initialize and set up the MCU GPIO pin as an input (connected to the INT1 pin of the sensor)
- In the sensor, set bit **SLP_ MODE_SEL** in **CTRL3** register (22h)
- In the sensor, set bit **INT1_DRDY** in **CTRL4_INT1_PAD_CTRL** register (23h)
- Set sensor's **ODR to 50 Hz** (recommended) using **ODR[3:0]** bits and operating mode to single data conversion using **MODE[1:0]** in **CTRL1** register (20h)
- Set bit **INTERRUPTS_ENABLE** in **CTRL7** register (3Fh)

Figure 1: Setting up single data conversion over an I²C/SPI command



Pseudocode – single data conversion over an I²C/SPI command

```
void AppSensorDataReq(void)     /* I2C/SPI command */
{
  write_reg(0x22, 0x03);        /* CTRL3(22h): Set SLP_MODE_1 */
}

void LIS2DW12_INT1_handler(void) /* INT1 DRDY Handler */
{
  printAxes();
}

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 */

  /* Manually controlled single data conversion enable */
```

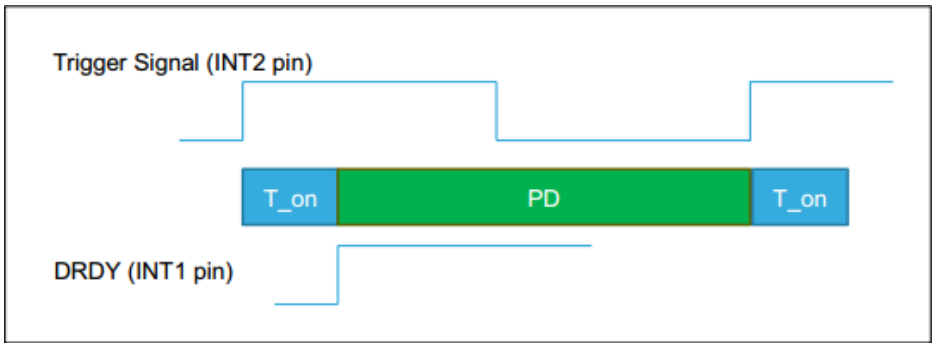
```
  write_reg(0x22, 0x02);        /* CTRL3(22h): Enable Single data conversion */
  write_reg(0x23, 0x01);        /* CTRL4_INT1_PAD_CTRL(23h): Data-ready routed to
INT1 */

  /* Start sensor */
  write_reg(0x20, 0x48);        /* CTRL1(20h): ODR 50Hz, Single data mode */
  delay(20);                    /* Settling time ( 1 sample, i.e. 1/ODR ) */
  write_reg(0x3f, 0x20);        /* CTRL7(3Fh): Enable interrupts */

  while (1)
  {
    /* … */
  }
}
```

Setting up single data conversion controlled by the sensor's INT2 pin working as an input

If the SLP_MODE_SEL bit in CTRL3 (22h) is set to 0, the application MCU can start the single data sensor conversion by setting high its GPIO (configured as an output) connected to the sensor's INT2 pin (configured as an input), as illustrated in Figure 3. When set, the sensor goes immediately from power-down mode to data conversion mode. When sensor data are available in the sensor's output registers, the SLP_MODE_1 bit is automatically cleared and the device is ready for another trigger from the MCU.

There are two ways to wait for data availability in the sensor's output data registers. The application can either configure the data-ready interrupt - DRDY (as shown in the pseudocode and in the time graph in Figure 2) or the application can continuously poll the DRDY bit in the STATUS (27h) register by an I²C/SPI read. The hardware connections are illustrated in Figure 3.

Figure 2: Single data conversion triggered by sensor's INT2 pin working as an input



To enable single data conversion controlled by the INT2 pin working as an input trigger, the application has to:

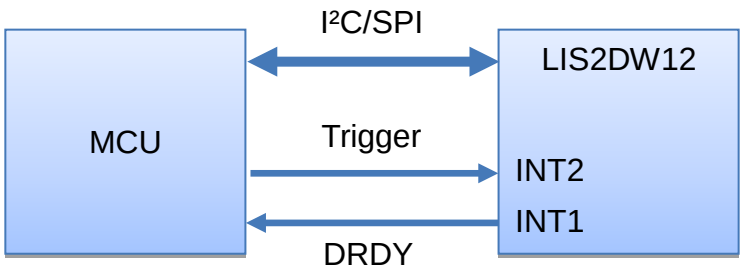
- Initialize the MCU
- Initialize and set up the MCU GPIO pin as an input (connected to the INT1 pin of the sensor)
- Initialize and set up the MCU GPIO pin as an output (connected to the INT2 pin of the sensor)
- In the sensor, initialize and set the INT2 pin as an input
- Disable bit **SLP_ MODE_SEL** in **CTRL3** register (22)
- Set bit **INT1_DRDY** in **CTRL4_INT1_PAD_CTRL** register (23h)
- Set sensor's **ODR to 50 Hz** (recommended) using **ODR[3:0]** bits and operating mode to single data conversion using **MODE[1:0]** in **CTRL1** register (20h)
- Set bit **INTERRUPTS_ENABLE** in **CTRL7** register (3Fh)

Pseudocode – single data conversion controlled by the sensor's INT2 pin working as an input

```
void LIS2DW12_INT1_handler(void)/* INT1 DRDY Handler */
{
  printAxes();
}

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

  /* Pin initialization */
  pin_Init(INT2_PIN);
  pin_Write(INT2_PIN, 0);

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

  /* Single data conversion enable */
  write_reg(0x22, 0x00);       /* CTRL3(22h): Enable Single data controlled by INT2
*/
  write_reg(0x23, 0x01);       /* CTRL4_INT1_PAD_CTRL(23h): Data-ready routed to
INT1 */

  /* Start sensor */
  write_reg(0x20, 0x48);       /* CTRL1(20h): Set ODR 50Hz, Single data mode */
  delay(20);                   /* Settling time ( 1 sample, i.e. 1/ODR ) */
```

```
  write_reg(0x3f, 0x20);       /* CTRL7(3Fh): Enable interrupts */

  while (1)
  {
    pinToggle(INT2_PIN, 300);  /* Toggle pin every 300ms (as example) */
  }
}
```

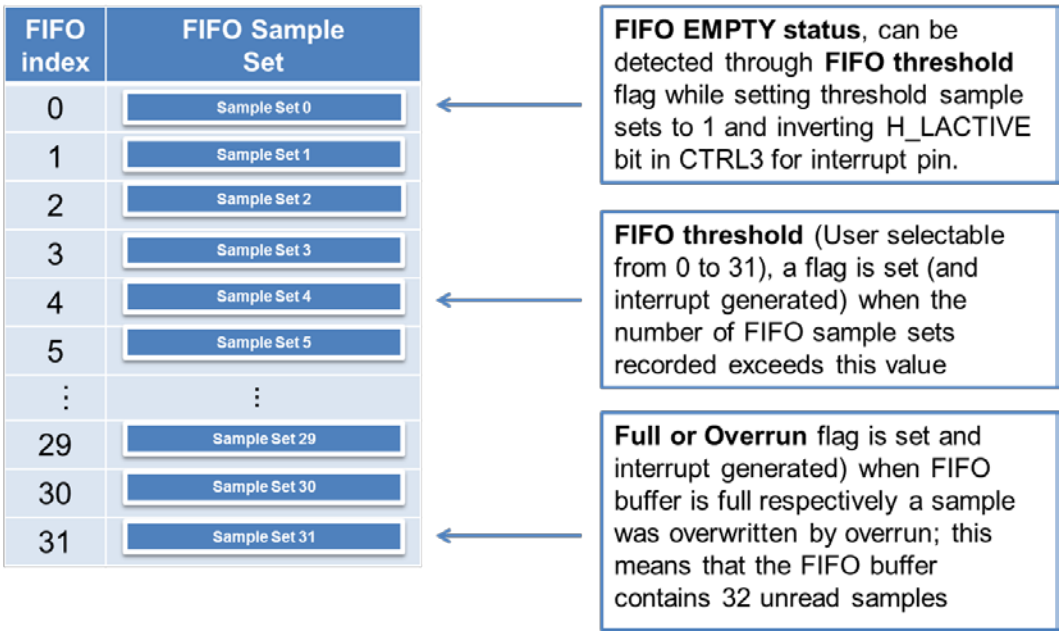
Figure 3: Single data conversion triggered by the sensor's INT2 pin working as an input



Single data conversion with data storage in FIFO

In order to limit interaction between the MCU and the sensor, it is possible to utilize the embedded LIS2DW12 FIFO buffer even in single data conversion mode. In this way it is possible to store up to 32 records per accelerometer axis and to read them in one-shot only when needed by the MCU application. There are also additional functions linked to the FIFO in order to generate an interrupt to the MCU according to the level of records in the FIFO, as seen in Figure 4. More information about the FIFO set-up can be found in the LIS2DW12 application note AN5038.

Figure 4: FIFO utilization with single data conversion mode



Setting up single data conversion over an I²C/SPI command with data storage in FIFO

To enable single data conversion over an I²C/SPI command with data storage in FIFO, the application has to:

- Initialize the MCU
- Initialize and set up the MCU GPIO pin as an input (connected to the INT1 pin of the sensor)
- Set FIFO mode with **FMode[2:0]** bits in **FIFO_CTRL** register (2Eh)
- In the sensor, set bit **SLP_ MODE_SEL** in **CTRL3** register (22h)
- Set bit **INT1_DIFF5** in **CTRL4_INT1_PAD_CTRL** register (23h)
- Set the sensor's **ODR to 50 Hz** (recommended) using **ODR[3:0]** bits and operating mode to single data conversion using **MODE[1:0]** in **CTRL1** register (20h)
- Set bit **INTERRUPTS_ENABLE** in **CTRL7** register (3Fh)

Pseudocode – single data conversion over an I²C/SPI command with data storage in FIFO

```
void AppSensorDataReq(void)     /* I2C/SPI command */
{
  write_reg(0x22, 0x03);        /* CTRL3(22h): Set SLP_ MODE_1 */
}

void LIS2DW12_INT1_handler(void) /* INT1 FIFO full Handler */
```

```
{
  print("Printing FIFO content\r\n");
  read_reg(0x2f, &fifo);        /* Get number of unread samples */
  fifo &= 0x3f;

  while (fifo != 0x00)          /* Print axes till FIFO is empty */
  {
    printAxes();
    read_reg(0x2f, &fifo);      /* Get number of unread samples */
    fifo &= 0x3f;
  }

  /* Reset FIFO */
  write_reg(0x2e, 0x00);
  write_reg(0x2e, 0x20);
}

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 */
  write_reg(0x2e, 0x20);        /* FIFO_CTRL(2Eh): FIFO mode enabled */

  /* Manually controlled single data conversion enable */
  write_reg(0x22, 0x02);        /* CTRL3(22h): Enable Single data conversion */
  write_reg(0x23, 0x04);        /* CTRL4_INT1_PAD_CTRL(23h): FIFO full routed to
INT1 */

  /* Start sensor */
  write_reg(0x20, 0x48);        /* CTRL1(20h): ODR 50Hz, Single data mode */
  delay(20);                    /* Settling time ( 1 sample, i.e. 1/ODR ) */
  write_reg(0x3f, 0x20);        /* CTRL7(3Fh): Enable interrupts */

  while (1)
  {
   /* … */
  }
}
```

Setting up single data conversion controlled by the sensor's INT2 pin working as an input with data storage in FIFO

To enable single data conversion controlled by the INT2 pin working as an input trigger with data storage in FIFO, the application has to:

- Initialize the MCU
- Initialize and set up the MCU GPIO pin as an input (connected to the INT1 pin of the sensor)
- Initialize and set up the MCU GPIO pin as an output (connected to the INT2 pin of the sensor)
- In the sensor, initialize and set the INT2 pin as an input
- Set FIFO mode with **FMode[2:0]** bits in **FIFO_CTRL** register (2Eh)
- Disable bit **SLP_ MODE_SEL** in **CTRL3** register (22h)
- Set bit **INT1_DIFF5** in **CTRL4_INT1_PAD_CTRL** register (23h)
- Set the sensor's **ODR to 50 Hz** (recommended) using **ODR[3:0]** bits and operating mode to single data conversion using **MODE[1:0]** in **CTRL1** register (20h)
- Set bit **INTERRUPTS_ENABLE** in **CTRL7** register (3Fh)

Pseudocode – single data conversion controlled by the sensor's INT2 pin working as an input with data storage in FIFO

```
void LIS2DW12_INT1_handler(void) /* INT1 FIFO full handler */
{
  print("Printing FIFO content\r\n");
  read_reg(0x2f, &fifo);        /* Get number of unread samples */
  fifo &= 0x3f;

  while (fifo != 0x00)          /* Print axes till FIFO is empty */
  {
    printAxes();
    read_reg(0x2f, &fifo);      /* Get number of unread samples */
    fifo &= 0x3f;
  }

  /* Reset FIFO */
  write_reg(0x2e, 0x00);
  write_reg(0x2e, 0x20);
}

int main(void)
{
```

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

  /* Pin initialization */
  pin_Init(INT2_PIN);
  pin_Write(INT2_PIN, 0);

  /* Initialization of sensor */
  write_reg(0x21, 0x08);        /* CTRL2(21h): Enable BDU */
  write_reg(0x25, 0x00);        /* CTRL6(25h): Set Full-scale to +/-2g */
  write_reg(0x2e, 0x20);        /* FIFO_CTRL(2Eh): FIFO mode enabled */

  /* Single data conversion enable */
  write_reg(0x22, 0x00);        /* CTRL3(22h): Enable Single data controlled by INT2
*/
  write_reg(0x23, 0x04);        /* CTRL4_INT1_PAD_CTRL(23h): FIFO full routed to
INT1 */

  /* Start sensor */
  write_reg(0x20, 0x48);        /* CTRL1(20h): Set ODR 50Hz, Single data mode */
  delay(20);                    /* Settling time ( 1 sample, i.e. 1/ODR ) */
  write_reg(0x3f, 0x20);        /* CTRL7(3Fh): Enable interrupts */

  while (1)
  {
    pin_Toggle(INT2_PIN, 300);  /* Toggle pin every 300ms (as example) */
  }
}
```

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
Documentation
Datasheet LIS2DW12, High-performance ultra-low-power 3-axis "femto" accelerometer
Application note, AN5038, LIS2DW12: always-on 3D accelerometer

Revision history

Date	Version	Changes
23-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.