

BMI08x FIFO usage

Bosch Sensortec



BOSCH

Invented for life

Application note: BMI08x FIFO usage

Document revision 1.1

Document release date November 2018

Document number BST-MIS-AN005-01

Technical reference code(s) 0 273 141 365, 0 273 141 366

Notes **Data and descriptions within this document are subject to change without notice.**

Product photos and pictures are for illustration purposes only and may differ from the real product's appearance.

Index of Contents

| | |
|--|-----------|
| 2. FIFO OPERATION | 4 |
| 2.1 FIFO OPERATING MODES | 4 |
| 2.2 FIFO INTERRUPTS | 4 |
| 3. ACCELEROMETER SENSOR FIFO BUFFER | 4 |
| 3.1 ENABLING FIFO AND SELECTING THE MODE | 4 |
| 3.1.1 MODE SELECTION | 4 |
| 3.1.2 FIFO DATA SAMPLING RATE..... | 4 |
| 3.1.3 FIFO SYNCHRONIZATION WITH EXTERNAL INTERRUPTS (TAG APPLICATION) FOR THE ACCEL | 4 |
| 3.2 DATA FORMAT IN FIFO..... | 5 |
| 3.2.1 ACCELERATION SENSOR DATA FRAME | 5 |
| 3.2.2 SKIP FRAME | 5 |
| 3.2.3 SENSORTIME FRAME | 5 |
| 3.2.4 FIFO INPUT CONFIG FRAME..... | 5 |
| 3.2.5 SAMPLE DROP FRAME..... | 6 |
| 3.2.6 FIFO PARTIAL FRAME READS AND OVERREADS..... | 6 |
| 3.3 FIFO INTERRUPTS | 6 |
| 3.4 FIFO RESET..... | 7 |
| 3.5 REGISTER MAP EXCERPT ACCELEROMETER | 8 |
| 3.5.1 OVERVIEW | 8 |
| 3.5.2 REGISTER 0x24-0x25: FIFO_LENGTH | 8 |
| 3.5.3 REGISTER 0x26: FIFO_DATA | 8 |
| 3.5.4 REGISTER 0x45: FIFO_DOWNS..... | 8 |
| 3.5.5 REGISTER 0x46 – 0x47: FIFO_WTM..... | 9 |
| 3.5.6 REGISTER 0x48: FIFO_CONFIG_0..... | 9 |
| 3.5.7 REGISTER 0x49: FIFO_CONFIG_1..... | 9 |
| 3.5.8 REGISTER 0x58: INT1_INT2_MAP_DATA | 10 |
| 4. GYROSCOPE SENSOR FIFO BUFFER | 11 |
| 4.1 ENABLING FIFO AND SELECTING THE MODE | 11 |
| 4.1.1 FIFO DATA SAMPLING RATE..... | 11 |
| 4.1.2 FIFO SYNCHRONIZATION WITH EXTERNAL INTERRUPTS (TAG APPLICATION) FOR THE GYROSCOPE | 11 |
| 4.2 FIFO DATA READOUT | 12 |
| 4.2.1 INTERFACE SPEED REQUIREMENTS FOR GYROSCOPE FIFO USE..... | 13 |
| 4.3 FIFO FRAME COUNTER AND OVERRUN FLAG | 13 |
| 4.4 FIFO INTERRUPTS | 13 |
| 4.5 REGISTER MAP EXCERPT GYROSCOPE..... | 14 |
| 4.5.1 OVERVIEW | 14 |
| 4.5.2 REGISTER 0x0A: INT_STATUS_1 | 14 |



4.5.3 REGISTER 0x0E: FIFO_STATUS 14

4.5.4 REGISTER 0x15: GYRO_INT_CTRL 15

4.5.5 REGISTER 0x18: INT3_INT4_IO_MAP 15

4.5.6 REGISTER 0x1E: FIFO_WM_ENABLE 15

4.5.7 REGISTER 0x34: FIFO_EXT_INT_S 16

4.5.8 REGISTER 0x3D: FIFO_CONFIG_0 16

4.5.9 REGISTER 0x3E: FIFO_CONFIG_1 16

4.5.10 REGISTER 0x3F: FIFO_DATA 17

5. PSEUDO CODE EXAMPLES 18

5.1 ACCELEROMETER FIFO READ EXAMPLE 18

5.2 GYROSCOPE FIFO READ EXAMPLE 19

6. LEGAL DISCLAIMER 21

6.1 ENGINEERING SAMPLES 21

6.2 PRODUCT USE 21

6.3 APPLICATION EXAMPLES AND HINTS 21

7. DOCUMENT HISTORY AND MODIFICATION 22

2. FIFO operation

The BMI08x sensor family offer two integrated FIFO buffers (FIFO = First in, first out) for accelerometer and gyroscope sensor signals, helping the user to reduce or even omit time critical read access to the sensor in order to obtain data with a high timing precision.

2.1 FIFO operating modes

The FIFO can be operated in different modes: FIFO (or stop-at-full) mode and STREAM mode.

- **FIFO or stop-at-full mode:** In FIFO or stop-at-full mode, the sensor values are stored in the FIFO buffer subsequently until it is full.
- **STREAM mode:** The STREAM mode works like the FIFO mode with the difference that once the buffer is full, the oldest data in the FIFO will be overwritten with the newest data from the sensor.

2.2 FIFO interrupts

The FIFO buffers support two different types of interrupts:

- **Watermark interrupt:** Triggered, when the fill level of the FIFO buffer reaches a user-defined level.
- **FIFO-full interrupt:** Triggered, when the FIFO is full.

3. Accelerometer sensor FIFO buffer

The accelerometer part of BMI088 has an integrated 1024 byte data FIFO. The FIFO captures data from the data registers in frames, and each frame contains only one sample of a sensor.

3.1 Enabling FIFO and selecting the mode

The FIFO for accelerometer sensor data is enabled by setting bit #6 in register 0x49 (see 3.5.7).

3.1.1 Mode selection

When STREAM mode is desired, than the bit #0 in register 0x48 has to be cleared (set to '0') (default value on power up reset, see 3.5.6).

For FIFO or stop-at-full mode, bit #0 has to be set to '1' in register 0x48.

3.1.2 FIFO data sampling rate

The input data rate to the FIFO is the same as the configured ODR of the sensor. However, it can be reduced selecting a down-sampling factor of 2^k with $k=[0, 1, \dots 7]$. The factor k must be written to bits #4-6 of register 0x45 (see 3.5.4).

3.1.3 FIFO synchronization with external interrupts (tag application) for the accel

If the INT1 and/or INT2 pin is configured as input pin (by setting `int2_io` in register `INT2_IO_CTRL` and/or setting `int1_io` in register `INT1_IO_CTRL`), signals on these pins can also be recorded in the FIFO, and the frames are "tagged" accordingly. Therefore the pins need to be activated for FIFO recording in register `FIFO_CONFIG_1` (see 3.5.7).

3.2 Data format in FIFO

The FIFO captures data in frames. The first byte is a header byte, defining the type of frame. From this, the number of consecutive bytes and their content can be derived.

The header byte consist of the header signature (first 6 bits) and two bits indicating the status of the interrupt pins INT1 and INT2 if configured accordingly (see 3.1.3).

3.2.1 Acceleration sensor data frame

- Frame length: 7 bytes (1 byte header + 6 bytes payload)
- Header:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|------------|------------|
| | 1 | 0 | 0 | 0 | 0 | 1 | [INT2 tag] | [INT1 tag] |

- Payload: the next bytes contain the sensor data in the same order as defined in the register map (addresses 0x12 – 0x17).

3.2.2 Skip Frame

In the case of FIFO overflows, in both FIFO and STREAM mode, a skip_frame is prepended to the FIFO content, when read out next time. A skip frame does not consume memory in the FIFO.

- Frame length: 2 bytes (1 byte header + 1 byte payload)
- Header:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|----------|----------|
| | 0 | 1 | 0 | 0 | 0 | 0 | reserved | reserved |

- Payload: one byte containing the number of skipped frames. When more than 0xFF frames have been skipped, 0xFF is returned.

3.2.3 Sensortime Frame

A sensortime frame is only sent if the FIFO becomes empty during the burst read. A sensortime frame does not consume memory in the FIFO.

- Frame length: 4 bytes (1 byte header + 3 bytes payload)
- Header:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|----------|----------|
| | 0 | 1 | 0 | 0 | 0 | 1 | reserved | reserved |

- Payload: Sensortime (content of registers 0x18 – 0x1A), taken when the last byte of the last frame is read.

3.2.4 FIFO input config Frame

Whenever the filter configuration or the range of the accelerometer sensor is changed, a FIFO input config frame is inserted into the FIFO, before the configuration change becomes active. E.g. when the bandwidth for the accelerometer filter is changed in Register ACC_CONF, a FIFO input

config frame is inserted before the first frame with accelerometer data with the new bandwidth configuration.

- Frame length: 2 bytes (1 byte header + 1 byte payload)
- Header:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|----------|----------|
| | 0 | 1 | 0 | 0 | 1 | 0 | reserved | reserved |

- Payload: The FIFO input config frame contains one byte of data, of which the following bits have a meaning (the content of the other bits can safely be ignored):
 - Bit #1: indicates that a configuration change through register ACC_RANGE becomes active (means for example that the range of the accelerometer was changed).
 - Bit #0: indicates that a configuration change through the registers ACC_CONF or FIFO_DOWNS becomes active (means of example that the filter settings were changed or the FIFO sampling rate was modified).

3.2.5 Sample drop Frame

After a reconfiguration, indicated by the fifo_Input_Config frame, the next sample may be dropped, until the sensor delivers valid data again. Instead, a drop frame is inserted at the ODR tick at which a sample was to be expected without reconfiguration.

- Frame length: 2 bytes (1 byte header + 1 byte payload)
- Header:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|----------|----------|
| | 0 | 1 | 0 | 1 | 0 | 0 | reserved | reserved |

- Payload: The sample drop frame contains one byte of data, whose content can be ignored.

3.2.6 FIFO partial frame reads and overreads

When a frame is only partially (uncompletely) read through the register FIFO_DATA it will be repeated completely with the next access. In the case of a FIFO overflow between the first partial read and the second read attempt, the frame may be deleted.

When more data is read from the FIFO than it contains valid data, 0x8000 is returned.

3.3 FIFO Interrupts

The FIFO supports two interrupts, a FIFO full interrupt and a watermark interrupt:

- The FIFO full interrupt is issued when the FIFO fill level is above the full threshold. The full threshold is reached just before the last two frames are stored in the FIFO.
- The FIFO watermark is issued when the FIFO fill level is superior or equal to the watermark level defined in register FIFO_WTM (see 3.5.5).

In order to enable/use the FIFO full or watermark interrupts they need to be mapped on the desired interrupt pin via INT1_INT2_MAP_DATA (see 3.5.8).



Both interrupts are suppressed when a read operation on the register FIFO_DATA is ongoing. Latched FIFO interrupts will only get cleared, if the status register gets read and the fill level is below the corresponding FIFO interrupt (full or watermark).

3.4 FIFO Reset

The user can trigger a FIFO reset by writing 0xB0 to ACC_SOFTRESET (register 0x7E).

3.5 Register map excerpt accelerometer

In addition to the register map as shown in the data sheet, the following registers are important for FIFO usage.

3.5.1 Overview

| Reg. Addr | Register Name | Reset Value | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 | |
|-----------|--------------------|-------------|------------------------|------------|-------------------------|------------|---------|------------|----------|------------|--|
| 0x58 | INT1_INT2_MAP_DATA | 0x00 | - | int2_drd_y | int2_fwm | int2_ffull | - | int1_drd_y | Int1_fwm | Int1_ffull | |
| 0x49 | FIFO_CONFIG_1 | 0x10 | - | acc_en | - | 1 | int1_en | int2_en | - | | |
| 0x48 | FIFO_CONFIG_0 | 0x02 | - | | | | | | 1 | mode | |
| 0x47 | FIFO_WTM_1 | 0x02 | - | | fifo_water_mark[12:8] | | | | | | |
| 0x46 | FIFO_WTM_0 | 0x00 | fifo_water_mark[7:0] | | | | | | | | |
| 0x45 | FIFO_DOWNS | 0x80 | 1 | fifo_downs | | - | | | | | |
| 0x26 | FIFO_DATA | 0x00 | fifo_data | | | | | | | | |
| 0x25 | FIFO_LENGTH_1 | 0x00 | - | | fifo_byte_counter[13:8] | | | | | | |
| 0x24 | FIFO_LENGTH_0 | 0x00 | fifo_byte_counter[7:0] | | | | | | | | |

3.5.2 Register 0x24-0x25: FIFO_LENGTH

The FIFO length registers FIFO_LENGTH_1 and FIFO_LENGTH_0 contain the 14 bit FIFO byte counter. The counter represents the current fill level of the FIFO buffer.

An empty FIFO corresponds to 0x8000. A FIFO content reset can be triggered by reading out all frames from the FIFO buffer or by writing 0xB0 into register 0x7E. The byte counter is updated when a complete frame is read or written.

3.5.3 Register 0x26: FIFO_DATA

When reading out data from FIFO, burst read access must be used. The address will not increase when burst read at the address of FIFO_DATA. The FIFO data is organized in frames as described in chapter 3.2.

When a frame is partially read through FIFO Data Register 0x26, it will be repeated completely with the next access. However, in case of a FIFO overflow between the first partial read and the second read attempt, the frame may be deleted.

3.5.4 Register 0x45: FIFO_DOWNS

| Bit | Name | Access | Reset value | Description |
|-------|-----------------|-----------|-------------|--|
| [7] | <i>Reserved</i> | <i>RW</i> | <i>0x01</i> | <i>This bit must always be '1'.</i> |
| [6:4] | fifo_downs | RO | 0x00 | Reduction of sample rate by a factor $2^{**}fifo_downs$. Example: fifo_downs=5 will reduce the FIFO sampling rate by a factor of $2^{**}5 = 32$ in relation to the chosen ODR of the sensor signal. |
| [3:0] | <i>reserved</i> | | | |

3.5.5 Register 0x46 – 0x47: FIFO_WTM

Registers containing the 13 bit FIFO watermark level value. A FIFO water mark interrupt signal is active if the FIFO fill level is equal or greater than `fifo_water_mark[12:0]` (unit of the fifo water mark is one byte).

3.5.6 Register 0x48: FIFO_CONFIG_0

Sets the FIFO mode.

| Bit | Name | Access | Reset value | Description | | | | | | |
|-------|-----------------|-----------|-------------|---|------|---------------|------|-------------|------|-----------|
| [7:2] | | | | <i>reserved</i> | | | | | | |
| [1] | <i>Reserved</i> | <i>RW</i> | <i>0x01</i> | <i>This bit must always be '1'.</i> | | | | | | |
| [0] | mode | <i>RW</i> | <i>0x00</i> | This parameter sets the measurement range. Possible values: <table border="1" data-bbox="801 788 1353 922" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>mode</th> <th>Range setting</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>STREAM mode</td> </tr> <tr> <td>0x01</td> <td>FIFO mode</td> </tr> </tbody> </table> | mode | Range setting | 0x00 | STREAM mode | 0x01 | FIFO mode |
| mode | Range setting | | | | | | | | | |
| 0x00 | STREAM mode | | | | | | | | | |
| 0x01 | FIFO mode | | | | | | | | | |

3.5.7 Register 0x49: FIFO_CONFIG_1

Selects sources for the FIFO buffer.

| Bit | Name | Access | Reset value | Description |
|-------|-----------------|-----------|-------------|--|
| [7] | | | | <i>reserved</i> |
| [6] | Acc_en | <i>RW</i> | <i>0x00</i> | Enables storing of accelerometer sensor data |
| [5] | <i>Reserved</i> | <i>RW</i> | <i>0x00</i> | |
| [4] | <i>Reserved</i> | <i>RW</i> | <i>0x01</i> | <i>This bit must always be '1'.</i> |
| [3] | Int1_en | <i>RW</i> | <i>0x00</i> | Enables storing of captured interrupt events at pin INT1 (pin needs to be configured as input pin accordingly) |
| [2] | Int2_en | <i>RW</i> | <i>0x00</i> | Enables storing of captured interrupt events at pin INT2 (pin needs to be configured as input pin accordingly) |
| [1:0] | <i>Reserved</i> | <i>RW</i> | <i>0x00</i> | |

3.5.8 Register 0x58: INT1_INT2_MAP_DATA

Maps various interrupts to the interrupt pins.

| Bit | Name | Access | Reset value | Description |
|-----|-----------------|--------|-------------|--|
| [7] | <i>reserved</i> | | | |
| [6] | int2_drdy | RW | 0x00 | Map data ready interrupt to pin INT2 |
| [5] | int2_fwm | RW | 0x00 | Map FIFO watermark interrupt to pin INT2 |
| [4] | int2_ffull | RW | 0x00 | Map FIFO full interrupt to pin INT2 |
| [3] | <i>reserved</i> | | | |
| [2] | int1_drdy | RW | 0x00 | Map data ready interrupt to pin INT1 |
| [1] | int1_fwm | RW | 0x00 | Map FIFO watermark interrupt to pin INT1 |
| [0] | int1_ffull | RW | 0x00 | Map FIFO full interrupt to pin INT1 |

4. Gyroscope sensor FIFO buffer

The gyroscope part of BMI088 features an integrated FIFO memory capable of storing up to 100 frames of data in FIFO mode. Each frame consists of three 16-bit rate_x,y,z data words, and 16 bits of interrupt data sampled at the same point in time.

4.1 Enabling FIFO and selecting the mode

The FIFO for gyroscope sensor data is enabled by setting the appropriate FIFO mode in Register 0x3E: FIFO_CONFIG_1.

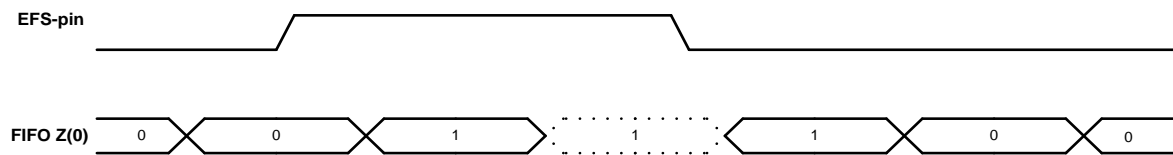
4.1.1 FIFO data sampling rate

The input data rate to the FIFO is the same as the configured ODR of the sensor.

4.1.2 FIFO synchronization with external interrupts (tag application) for the gyroscope

The FIFO of the gyroscope features a mode that allows the precise synchronization of external events with the gyroscope angular rate saved in the FIFO. This synchronization can be used for example for image and video stabilization applications.

Any of the gyroscope interrupt pins (INT3 or INT4) can be reconfigured to act as input pin, but not both. In addition, the tag mode has to be enabled. The so configured interrupt pin will then behave as an input pin and not as an interrupt pin. The working principle is shown in below figure:



Timing diagram for external FIFO synchronization. EFS-pin is the Interrupt pin configured to capture external events. FIFO z(0) is the least significant bit of the z-axis gyro data stored in the FIFO.

In order to enable the tag mode, bit 5 must be set in register 0x34 (see 0). The pin can be chosen in the same register, bit 4.

In this mode, the least significant bit of the z-axis is used as tag-bit, therefore losing its meaning as gyroscope data bit. The remaining 15 bits of the z-axis gyroscope data keep the same meaning as in standard mode.

Once the pin, which is configured for the tag mode, is set to high level, the next FIFO word will be marked with a tag (z-axis LSB = 1). While pin is kept at a high level, the corresponding FIFO words will continuously be tagged. After the pin is reset to low level, the immediate next FIFO word could still be tagged, and only after this word, the next tag will be reset (z-axis LSB=0). This is shown in the above diagram.

The tag synchronizes external events with the same time precision as the FIFO update rate. Therefore update rate of the tag is determined by the output data rate.



4.2 FIFO Data Readout

The FIFO stores the data that are also available at the read-out registers 0x02-0x07. Thus, all configuration settings apply to the FIFO data as well as the data readout registers. The FIFO read out is possible through

Register 0x3F: FIFO_DATA. The readout can be performed using burst mode. A single burst can read out one or more frames at a time. If a frame is not read completely due to an incomplete read operation, the remaining part of the frame is lost. In this case the FIFO aligns to the next frame during the next read operation.

The data format is described in 0.

4.2.1 Interface speed requirements for Gyroscope FIFO use

In order to use the FIFO effectively, larger blocks of data need to be read out quickly. Depending on the output data rate of the sensor, this can impose requirements on the interface.

The output data rate of the gyroscope is determined by the filter configuration (see the data sheet of the sensor). What interface speed is required depends on the selected rate.

- For an I²C speed of 400 kHz, every filter mode can be used.
- For an I²C speed of 200 kHz, only modes with an output data rate of 1 KHz and below are recommended.
- For an I²C speed of 100 kHz, only modes with an output data rate of 400 Hz and below are recommended.

4.3 FIFO Frame Counter and Overrun Flag

The frame counter (address 0x0E bits<6:0>, see 4.5.3) indicates the current fill level of the buffer. If additional frames are written to the buffer although the FIFO is full, the overrun flag (register 0x0E bit 7) is set. If the FIFO is reset, the FIFO fill level indicated in the frame_counter<6:0> is set to '0' and the overrun flag is reset each time a write operation happens to the FIFO configuration registers.

Note: the overrun bit is not reset when the FIFO fill level frame_counter<6:0> has decremented to '0' due to reading from the FIFO_DATA register, but only when a write operation is performed on FIFO configuration registers.

4.4 FIFO Interrupts

The FIFO supports two interrupts, a FIFO full interrupt and a watermark interrupt:

- The **FIFO full interrupt** is issued when the buffer has been fully filled with samples. In FIFO mode this occurs after 100 samples, and in STREAM mode after 99 samples, have been stored in a previously empty FIFO. The status of the FIFO-full interrupt may be read back through the status bit in INT_STATUS_1 register 0x0A (see 4.5.2).
- The **watermark interrupt** is issued when the fill level in the buffer has reached the frame number defined by the water mark level trigger in 0x3D. The status of the watermark may be read back through the address 0x0A bit 4 (fifo_int) status bit (see 4.5.2). Writing to water mark level trigger in register 0x3D clears the FIFO buffer.

4.5 Register map excerpt gyroscope

4.5.1 Overview

| Reg. Addr. | Register name | Reset value | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|------------|------------------|-------------|---------------------------|--------------------------------------|---------------|----------------|-----------|------|-----------|------|
| 0x3F | FIFO_DATA | N/A | fifo_data_output_register | | | | | | | |
| 0x3E | FIFO_CONFIG_1 | 0x00 | fifo_mode | | - | | | | | |
| 0x3D | FIFO_CONFIG_0 | 0x00 | - | fifo_water_mark_level_trigger_retain | | | | | | |
| 0x34 | FIFO_EXT_INT_S | 0x00 | - | | ext_fifo_s_en | ext_fifo_s_sel | - | | | |
| 0x1E | FIFO_WM_EN | 0x00 | fifo_watermark_enable | | | | | | | |
| 0x18 | INT3_INT4_IO_MAP | 0x00 | Int4_data | - | Int4_fifo | - | Int3_fifo | - | Int3_data | |
| 0x15 | GYRO_INT_CTRL | 0x00 | data_en | fifo_en | - | | | | | |
| 0x0E | FIFO_STATUS | N/A | fifo_overrun | fifo_frame_counter | | | | | | |
| 0x0A | INT_STATUS_1 | N/A | data_int | - | fifo_int | - | | | | |

4.5.2 Register 0x0A: INT_STATUS_1

The register contains interrupt status information.

| Bit | Name | Access | Reset value | Description |
|-------|-----------------|--------|-------------|---------------------------|
| [7] | data_int | RO | N/A | New data interrupt status |
| [6:5] | <i>reserved</i> | | | |
| [4] | fifo_int | RO | N/A | FIFO interrupt status |
| [3:0] | <i>reserved</i> | | | |

4.5.3 Register 0x0E: FIFO_STATUS

The register contains FIFO status information.

| Bit | Name | Access | Reset value | Description |
|-------|--------------------|--------|-------------|--|
| [7] | Fifo_overrun | RO | N/A | If set, FIFO overrun condition has occurred. Note: flag can only be cleared by writing to the FIFO configuration register FIFO_CONFIG_1 |
| [6:0] | Fifo_frame_counter | RO | N/A | Current fill level of FIFO buffer. An empty FIFO corresponds to 0x00. The frame counter can be cleared by reading out all frames from the FIFO buffer or writing to the FIFO configuration register FIFO_CONFIG_1. |

4.5.4 Register 0x15: GYRO_INT_CTRL

In this register the data ready interrupt and FIFO interrupt can be enabled.

| Bit | Name | Access | Reset value | Description |
|-------|-----------------|--------|-------------|---|
| [7] | data_en | RW | 0x0 | Enables the new data interrupt to be triggered on new data. |
| [6] | fifo_en | RW | 0x0 | Enables the FIFO interrupt. |
| [5:0] | <i>reserved</i> | | | |

4.5.5 Register 0x18: INT3_INT4_IO_MAP

Map the data ready interrupt and FIFO interrupt to one of the interrupt pins INT3 and/or INT4.

| Bit | Name | Access | Reset value | Description |
|-------|-----------------|--------|-------------|---|
| [7] | Int4_data | RW | 0x0 | Data ready interrupt is mapped to INT4 pin. |
| [6] | <i>reserved</i> | | | |
| [5] | Int4_fifo | RW | 0x0 | FIFO interrupt is mapped to INT4. |
| [4:3] | <i>reserved</i> | | | |
| [2] | Int3_fifo | RW | 0x0 | FIFO interrupt is mapped to INT3. |
| [1] | <i>reserved</i> | | | |
| [0] | Int3_data | RW | 0x0 | Data ready interrupt is mapped to INT3 pin. |

4.5.6 Register 0x1E: FIFO_WM_ENABLE

| Bit | Access | Reset value | Description | | | | | | |
|----------------|---|-------------|---|----------------|-------------|------|---|------|--|
| [7:0] | RW | 0x08 | Enables FIFO watermark level interrupt. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>fifo_wm_enable</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>0x08</td> <td>FIFO watermark level interrupt disabled</td> </tr> <tr> <td>0x88</td> <td>FIFO watermark level interrupt enabled</td> </tr> </tbody> </table> | fifo_wm_enable | description | 0x08 | FIFO watermark level interrupt disabled | 0x88 | FIFO watermark level interrupt enabled |
| fifo_wm_enable | description | | | | | | | | |
| 0x08 | FIFO watermark level interrupt disabled | | | | | | | | |
| 0x88 | FIFO watermark level interrupt enabled | | | | | | | | |

4.5.7 Register 0x34: FIFO_EXT_INT_S

| Bit | Name | Access | Reset value | Description | | | | | | |
|----------------|--------------------|--------|-------------|--|----------------|----------|-----|--------------------|-----|--------------------|
| [5] | ext_fifo_s_en | RW | 0x00 | It set, enables external FIFO synchronization mode | | | | | | |
| [4] | ext_fifo_s_sel | RW | 0x00 | selects source for external FIFO synchronization <table border="1" data-bbox="774 526 1404 689"> <thead> <tr> <th>ext_fifo_s_sel</th> <th>Behavior</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Source is pin INT3</td> </tr> <tr> <td>0x1</td> <td>Source is pin INT4</td> </tr> </tbody> </table> | ext_fifo_s_sel | Behavior | 0x0 | Source is pin INT3 | 0x1 | Source is pin INT4 |
| ext_fifo_s_sel | Behavior | | | | | | | | | |
| 0x0 | Source is pin INT3 | | | | | | | | | |
| 0x1 | Source is pin INT4 | | | | | | | | | |

4.5.8 Register 0x3D: FIFO_CONFIG_0

| Bit | Access | Reset value | Description |
|-------|--------|-------------|--|
| [6:0] | RW | 0x00 | fifo_water_mark_level_trigger_retain<6:0> defines the FIFO watermark level. An interrupt will be generated, when the number of entries in the FIFO exceeds fifo_water_mark_level_trigger_retain<6:0>. Writing to this register clears the FIFO buffer. |

4.5.9 Register 0x3E: FIFO_CONFIG_1

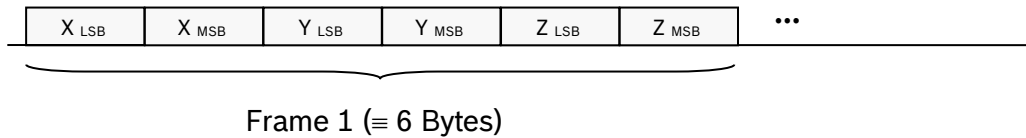
Contains FIFO configuration settings. The FIFO buffer memory is cleared and the fifo-full flag is cleared when writing to FIFO_CONFIG_1 register. In addition, the FIFO overrun flag (see 4.5.3) is cleared (it overrun occurred before).

| Bit | Access | Reset value | Description | | | | | | | | | | | | |
|-------------|--------|---|--|-----------|------|-------------|------|------|---|------|--------|---|-------------|--|-----------------|
| [7:0] | RW | 0x00 | <table border="1" data-bbox="550 1451 1404 1724"> <thead> <tr> <th>fifo_mode</th> <th>mode</th> <th>description</th> </tr> </thead> <tbody> <tr> <td>0x40</td> <td>FIFO</td> <td>data collection stops once buffer is full (i.e. filled with 100 frames)</td> </tr> <tr> <td>0x80</td> <td>STREAM</td> <td>sampling continues when buffer is full (i.e. filled with 99 frames); old is discarded</td> </tr> <tr> <td><i>else</i></td> <td></td> <td><i>reserved</i></td> </tr> </tbody> </table> | fifo_mode | mode | description | 0x40 | FIFO | data collection stops once buffer is full (i.e. filled with 100 frames) | 0x80 | STREAM | sampling continues when buffer is full (i.e. filled with 99 frames); old is discarded | <i>else</i> | | <i>reserved</i> |
| fifo_mode | mode | description | | | | | | | | | | | | | |
| 0x40 | FIFO | data collection stops once buffer is full (i.e. filled with 100 frames) | | | | | | | | | | | | | |
| 0x80 | STREAM | sampling continues when buffer is full (i.e. filled with 99 frames); old is discarded | | | | | | | | | | | | | |
| <i>else</i> | | <i>reserved</i> | | | | | | | | | | | | | |

4.5.10 Register 0x3F: FIFO_DATA

FIFO data readout register. The format of the LSB and MSB components corresponds to that of the angular rate data readout registers. Read burst access may be used since the address counter will not increment when the read burst is started at the address of FIFO_DATA. The entire frame is discarded when a frame is only partially read out.

The format of the data read-out from register 0x3F is as follows:



5. Pseudo code examples

The following examples are based on the sensorAPI (latest version can be found here: <https://github.com/BoschSensortec/BMI08x-Sensor-API>). They simply demonstrate the bare functionality. The examples were developed using the COINES framework, available on Bosch Sensortec's Website from September 2018 onwards. The user may deduce the functionality from the function calls shown in the examples below from the function names (for example, `coins_delay_msec(100)` obviously adds a delay of 100ms).

The examples make usage of the following support function:

```
static void unpack_sensor_data(struct bmi08x_sensor_data *sens_data, uint8_t *buffer)
{
    uint16_t data_lsb;
    uint16_t data_msb;
    uint16_t start_idx = 0;

    /* Gyro raw x data */
    data_lsb = buffer[start_idx++];
    data_msb = buffer[start_idx++];
    sens_data->x = (int16_t)((data_msb << 8) | data_lsb);

    /* Gyro raw y data */
    data_lsb = buffer[start_idx++];
    data_msb = buffer[start_idx++];
    sens_data->y = (int16_t)((data_msb << 8) | data_lsb);

    /* Gyro raw z data */
    data_lsb = buffer[start_idx++];
    data_msb = buffer[start_idx++];
    sens_data->z = (int16_t)((data_msb << 8) | data_lsb);
}
```

5.1 Accelerometer FIFO read example

In this example, the accelerometer part is initialized, FIFO stop-at-full mode is selected, and a watermark is set at 42 bytes.

```
struct bmi08x_sensor_data bmi08x_accel;
bmi08a_init(&bmi08xdev);

/* Switch on accelerometer */
bmi08xdev.accel_cfg.power = BMI08X_ACCEL_PM_ACTIVE;
bmi08a_set_power_mode(&bmi08xdev);
coins_delay_msec(50);

/* ODR = 1.6kHz, no oversampling */
bmi08xdev.accel_cfg.bw = BMI08X_ACCEL_BW_NORMAL;
bmi08xdev.accel_cfg.odr = BMI08X_ACCEL_ODR_400_HZ;
bmi08a_set_meas_conf(&bmi08xdev);
coins_delay_msec(100);

/* Desired FIFO mode is stop-at-full: set bit #0 to 1 in 0x48. Bit #1 must always be one! */
buffer[0] = 0x01 | 0x02;
rslt = bmi08a_set_regs(0x48, &buffer[0], 1, &bmi08xdev);

/* Downsampling factor 2**4 = 16: write 4 into bit #4-6 of reg. 0x45. Bit #7 must always be one! */
buffer[0] = (4 << 4) | 0x80;
rslt = bmi08a_set_regs(0x45, &buffer[0], 1, &bmi08xdev);

/* Set water mark to 42 bytes (aka 6 frames, each 7 bytes: 1 byte header + 6 bytes accel data) */
uint16_t wml = 42;
buffer[0] = (uint8_t) wml & 0xff;
```

```
buffer[1] = (uint8_t) (wml >> 8) & 0xff;
rslt = bmi08a_set_regs(0x46, &buffer[0], 2, &bmi08xdev);

/* Enable the actual FIFO functionality: write 0x50 to 0x49. Bit #4 must always be one! */
buffer[0] = 0x10 | 0x40;
rslt = bmi08a_set_regs(0x49, &buffer[0], 1, &bmi08xdev);
coins_delay_msec(50);

int fifo_fill_level = 0;
while(fifo_fill_level < wml)
{
    rslt = bmi08a_get_regs(0x24, buffer, 2, &bmi08xdev);
    fifo_fill_level = buffer[0] + 256 * buffer[1];
}

bmi08a_get_regs(0x26, buffer, fifo_fill_level, &bmi08xdev);
/* This is a super-simple FIFO parsing loop, hoping it will only find valid accel data packets */
for(int i = 0; i < fifo_fill_level; )
{
    /* Header of acceleration sensor data frame: 100001xxb, where x is INT1/INT2 tag, ignored here */
    if(buffer[i] == (0x84 & 0x8c))
    {
        unpack_sensor_data(&bmi08x_accel, &buffer[i + 1]);
        printf("frame: %03d ax:%d ay:%d az:%d\n", i/6, bmi08x_accel.x, bmi08x_accel.y, bmi08x_accel.z);
        i += 7;
    }
    else
        i++;
}
}
```

5.2 Gyroscope FIFO read example

In this example, the gyroscope part is initialized, FIFO stop-at-full mode is selected, FIFO interrupt is routed to pin INT3, the watermark is set to 8 frames.

```
struct bmi08x_sensor_data bmi08x_gyro;
bmi08g_init(&bmi08xdev);

/* Configure INT3 as output pin, level low: nothing to do, already default (reg. 0x16) */
/* Set INT3 as FIFO interrupt pin */
buffer[0] = 0x04; // Set bit #2
bmi08g_set_regs(0x18, &buffer[0], 1, &bmi08xdev);

/* Enable FIFO data interrupt */
bmi08g_get_regs(0x15, &buffer[0], 1, &bmi08xdev);
buffer[0] |= 0x40; // Set bit #6
bmi08g_set_regs(0x15, &buffer[0], 1, &bmi08xdev);

/* Enable watermark interrupt */
buffer[0] = 0x88;
bmi08g_set_regs(0x1e, &buffer[0], 1, &bmi08xdev);

/* Set watermark to 8 frames */
uint8_t watermark = 8;
bmi08g_set_regs(0x3d, &watermark, 1, &bmi08xdev);

/* Switch on FIFO mode (stop-at-full) */
buffer[0] = 0x40;
bmi08g_set_regs(0x3e, &buffer[0], 1, &bmi08xdev);

/* Wait for FIFO to be filled */
enum coins_pin_value pin = COINES_PIN_VALUE_LOW;
printf("FIFO frame counter (reg 0x0e & 0x7f): ");
while(pin == COINES_PIN_VALUE_LOW)
{
    bmi08g_get_regs(0x0e, buffer, 1, &bmi08xdev);
    printf("%d ... ", buffer[0] & 0x7f);
}
```



```
    coines_get_pin_config(COINES_SHUTTLE_PIN_22, (enum coines_pin_direction *) &buffer[0], (enum
coines_pin_value *) &pin);
}
printf("%d frames.\n", watermark);

/* read 48 bytes of aligned sensor data.
   Note: current COINES implementation does not allow to read more than 53 bytes! */
bmi08g_get_regs(0x3f, buffer, 8 * 6, &bmi08xdev);
for(int i = 0; i < 8; i ++){
    unpack_sensor_data(&bmi08x_gyro, &buffer[i * 6]);
    printf("frame: %03d gx:%d gy:%d gz:%d\n", i, bmi08x_gyro.x, bmi08x_gyro.y, bmi08x_gyro.z);
}
```

6. Legal disclaimer

6.1 Engineering samples

Engineering Samples are marked with an asterisk (*) or (e). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

6.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They are not designed or approved for use in military applications, life-support appliances, safety-critical automotive applications and devices or systems where malfunctions of these products can reasonably be expected to result in personal injury. They may only be used within the parameters of this product data sheet.

The resale and/or use of products are at the Purchaser's own risk and the Purchaser's own responsibility.

The Purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The Purchaser accepts the responsibility to monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of any security relevant incidents.

6.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.

7. Document history and modification

| Rev. No | Chapter | Description of modification/changes | Date |
|---------|---------|-------------------------------------|-------------|
| 1.1 | - | Document name change to MIS-AN005 | 28-Nov-2018 |
| 1.0 | - | Initial release | 27-Jul-2018 |
| | | | |

Bosch Sensortec GmbH
Gerhard-Kindler-Strasse 9
72770 Reutlingen / Germany

contact@bosch-sensortec.com
www.bosch-sensortec.com

Modifications reserved | Printed in Germany

Document number: BST-MIS-AN005-01
Revision_1_1_181128