**Sample Drop Frame**

A sample drop frame has always one byte payload, defined through

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

Sample drop frame will be inserted after a Fifo_Input_Config frame at the ODR tick at which the sample was dropped and only if no other sensor provides a valid sample at this ODR tick. If another sensor provides valid data, the data of this sensor is just not included and the appropriate header bit of the data frame is not set.

Sample drop frames will be inserted only for transition phases after configuration changes, not for samples dropped between sensor enable and first valid sample. For a detailed description of configuration changes see Section 4.5, Subsection “Configuration Changes”.

*FIFO Partial frame reads*

When a frame is only partially read through the Register FIFO_DATA it will be repeated completely with the next access both in headerless and in header mode. In headermode, this includes the header. In the case of a FIFO overflow between the first partial read and the second read attempt, the frame may be deleted.

*FIFO overreads*

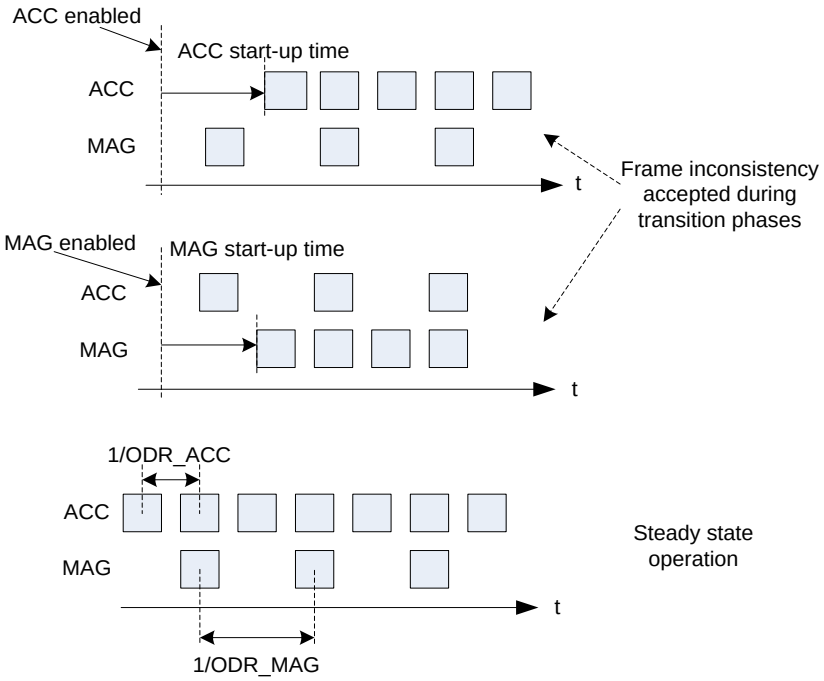
When more data are read from the FIFO than it contains valid data, 0x8000 is returned in headerless mode. While in header mode 0x0080 is returned, where 0x80 indicates an invalid frame.

### 4.6.3 FIFO data synchronization

All sensor data are sampled with respect to a common ODR time grid. Even if a different ODR is selected for the acceleration and the magnetic sensor the data remains synchronized:

If a frame contains a sample from a sensor element with ODR x, then it must contain also samples of all sensor elements with an ODR y>=x. This applies for steady state operation. In transition phases, it is more important not to lose data, therefore exceptions are possible if the sensor elements with ODR y>=x do not have data, e.g. due to a sensor configuration change.´

FIFO Data Synchronization Scheme in the following figure illustrates the steady state and transient operating conditions.


ACC enabled
ACC start-up time
ACC
MAG
t
Frame inconsistency
accepted during
transition phases
MAG enabled
MAG start-up time
ACC
MAG
t
1/ODR_ACC
ACC
MAG
t
Steady state
operation
1/ODR_MAG

### 4.6.4 FIFO synchronization with external interrupts

External interrupts may be synchronized into the FIFO data. For this operation mode the FIFO_CONFIG_1.fifo_tag_int1_en and/or FIFO_CONFIG_1.fifo_tag_int2_en need to be enabled, as well as INT1_IO_CTRL.input_en and/or INT2_IO_CTRL.input_en. The fh_ext field in FIFO header will then be set according to the signal at the INT1/INT2 inputs.

### 4.6.5 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 equal or above a watermark defined in Register FIFO_WTM_0 and FIFO_WTM_1.

In order to enable/use the FIFO full or watermark interrupts map them on the desired interrupt pin via INT_MAP_DATA.

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).

### 4.6.6 FIFO Flush

The user can trigger a FIFO reset by writing the command fifo_flash (0xB0) in CMD.
Automatic resets are only performed in the following cases:

- A sensor is enabled or disabled in headerless mode
- A transition between headerless and headermode or vice versa has occurred.
- Size of auxiliary sensor data in a frame changed in header or headerless mode

## 4.7 Integrated Features set:

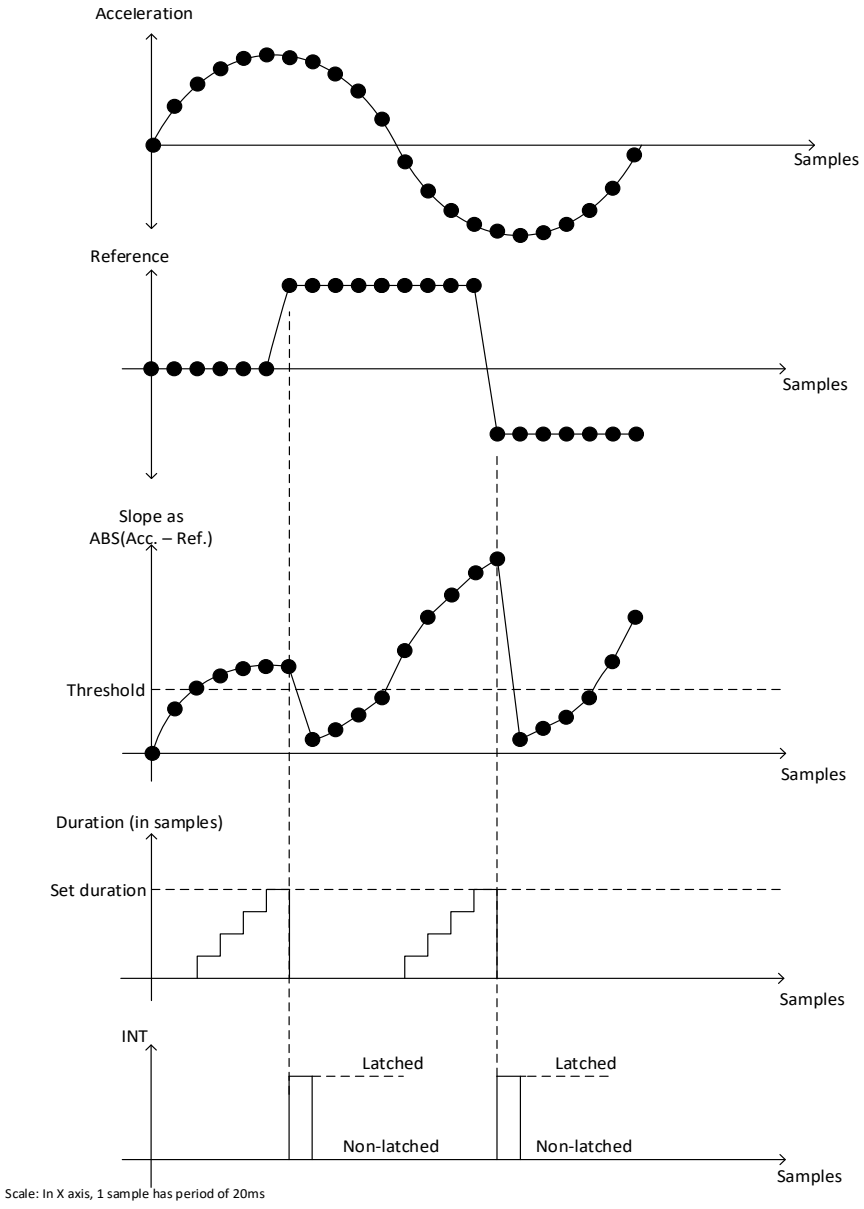
### 4.7.1 Any Motion / No motion detection

Any-motion detection:

Any-motion detection uses the slope between current input and reference acceleration samples to detect the motion status of the device. Feature can be enabled by setting at least one of the following: FEATURES_IN.any_motion.settings_2.x_en, FEATURES_IN.any_motion.settings_2.y_en and FEATURES_IN.any_motion.settings_2.z_en, respectively for each axis.

Any-motion provides an interrupt when the absolute value of the slope exceeds the configurable FEATURES_IN.any_motion.settings_1.threshold for consecutive FEATURES_IN.any_motion.settings_2.duration samples for at-least one of the enabled sensing axis. Reference acceleration sample is updated only when an any-motion interrupt is triggered. The interrupt status is reset as soon as the slope falls below the set FEATURES_IN.any_motion.setings_1.threshold value. The signals and timings relevant to the any-motion interrupt functionality are depicted in the figure below:

Signal and timing diagram for any-motion interrupt detection



Configuration settings:

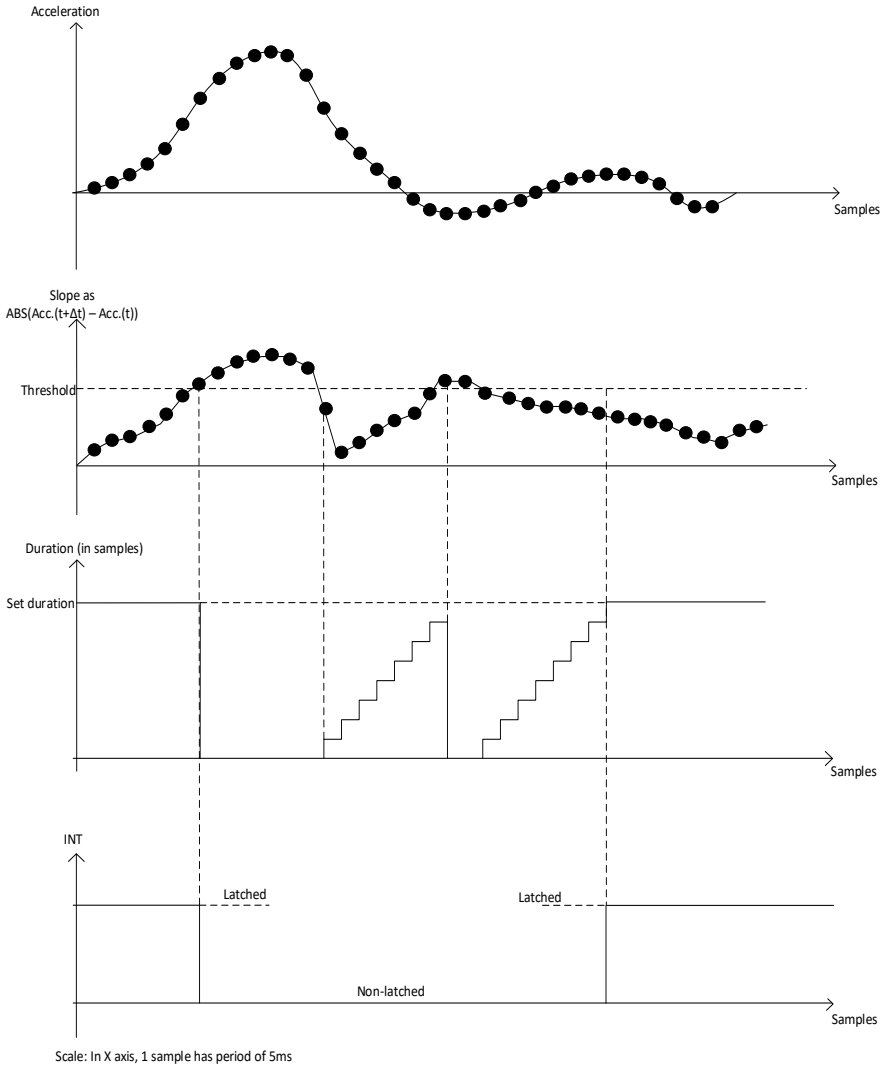
1. FEATURES_IN.any_motion.settings_1.threshold – the slope threshold.
2. FEATURES_IN.any_motion.settings_2.duration – the number of consecutive data points for which the threshold condition must be respected, for interrupt assertion.
3. FEATURES_IN.any_motion.settings_2.x_en – indicates if this feature is enabled for x axis
4. FEATURES_IN.any_motion.settings_2.y_en – indicates if this feature is enabled for y axis
5. FEATURES_IN.any_motion.settings_2.z_en –indicates if this feature is enabled for z axis

No Motion Detection:

No-motion detection uses the slope between two consecutive acceleration signal samples to detect static state of the device. Feature can be enabled by setting at least one of the following flags: FEATURES_IN.no_motion.settings_2.x_en, FEATURES_IN.no_motion.settings_2.y_en and FEATURES_IN.no_motion.settings_2.z_en, respectively for each axis.

No-motion interrupt is triggered when the slope on all enabled sensing axis remains smaller than the configurable FEATURES_IN.no_motion.settings_1.threshold for the duration configured by FEATURES_IN.no_motion.settings_2.duration. No-motion interrupt is cleared as soon as the acceleration slope exceeds the set threshold. The signals and timings relevant to the no-motion interrupt functionality are depicted in the figure below.

Signal and timing diagram for no-motion interrupt detection



Register FEATURES_IN.no_motion.settings_2.duration defines the number of consecutive data points for which the slope of enabled axis must be smaller than the threshold for an interrupt to be asserted.

Configuration settings:

1. FEATURES_IN.no_motion.settings_1.threshold – the slope threshold.
2. FEATURES_IN.no_motion.settings_2.duration – the number of consecutive data points for which the threshold condition must be respected, for interrupt assertion.
3. FEATURES_IN.no_motion.settings_2.x_en – indicates if this feature is enabled for x axis
4. FEATURES_IN.no_motion.settings_2.y_en – indicates if this feature is enabled for y axis
5. FEATURES_IN.no_motion.settings_2.z_en –indicates if this feature is enabled for z axis

**Note**: The firmware image with any motion and no motion feature set mentioned in the section above is available for download under the following link
https://github.com/BoschSensortec/BMA490L-Sensor-API

## 4.8 General Interrupt Pin configuration

### Electrical Interrupt Pin Behavior

Both interrupt pins INT1 and INT2 can be configured to show the desired electrical behavior.
Interrupt pins can be enabled in INT1_IO_CTRL.output_en respectively INT2_IO_CTRL.output_en.
The characteristic of the output driver of the interrupt pins may be configured with bits INT1_IO_CTRL.od and INT2_IO_CTRL.od. By setting these bits to 0b1, the output driver shows open-drive characteristic, by setting the configuration bits to 0b0, the output driver shows push-pull characteristic.
The electrical behavior of the Interrupt pins, whenever an interrupt is triggered, can be configured as either “active-high” or “active-low” via INT1_IO_CTRL.lvl respectively INT2_IO_CTRL.lvl.

Both interrupt pins can be configured as input pins via INT1_IO_CTRL.input_en respectively INT2_IO_CTRL.input_en. This is necessary when FIFO tag feature is used (see the respective FIFO chapter) If both are enabled, the input (e.g. marking FIFO) is driven by the interrupt output.
BMA490L supports edge and level triggered interrupt inputs, this can be configured through INT1_IO_CTRL.edge_ctrl respectively INT2_IO_CTRL.edge_ctrl.

BMA490L supports non-latched and latched interrupts modes for data-ready, FIFO full and FIFO watermark. The mode is selected by INT_LATCH.int_latch. The feature interrupts described in chapter FIFO Interrupts, support only latched mode described below.
In latched mode an asserted interrupt status in INT_STATUS_0 or INT_STATUS_1 and the selected pin are cleared if the corresponding status register is read. If more than one interrupt pin is used in latched mode, all interrupts in INT_STATUS_0 should be mapped to one pin and all interrupts in INT_STATUS_1 should be mapped to the other pin. If just one interrupt pin is used all interrupts may be mapped to this pin. If the activation condition still holds when it is cleared, the interrupt status is asserted again when the interrupt condition holds again.
In the non-latched mode (only for data-ready, FIFO full and FIFO watermark) the interrupt status bit and the selected pin are reset as soon as the activation condition is not valid anymore.

### Interrupt Pin Mapping

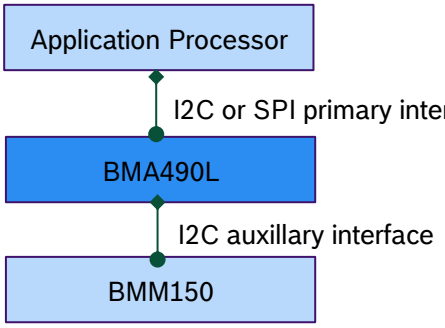
In order, for the Host to react to the features output, they can be mapped to the external pin INT1 or pin INT2, by setting the corresponding bits from the registers INT1_MAP, respectively INT2_MAP.

To disconnect the features outputs to the external pins, the same corresponding bits must be reset, from the registers, INT1_MAP, respectively INT2_MAP.

Once a feature triggered the output pin, the Host can read out the corresponding bit from the register, INT_STATUS_0 (Feature Interrupts) or INT_STATUS_1 (FIFO and data ready).

## 4.9 Auxiliary Sensor Interface

The auxiliary interface allows to attach one auxiliary sensor (e.g. magnetometer) on dedicated auxiliary sensor interface as shown below.




6 DOF Solution w/ BMA490L and BMM150

### 4.9.1 Structure and Concept

The BMA490L controls the data acquisition of the auxiliary sensor and presents the data to the application processor through the primary I2C or SPI interface. No other I2C master or slave devices must be attached to the auxiliary sensor interface.

The BMA490L autonomously reads the sensor data from a compatible auxiliary sensor without intervention of the application processor and stores the data in its data registers and FIFO. The initial setup of the auxiliary sensor after power-on is done through indirect addressing (in setup mode as described in following section).

The main benefits of the auxiliary sensor interface are

- Synchronization of sensor data of auxiliary sensor and accelerometer. This results in an improved sensor data fusion quality.
- Usage of the BMA490L FIFO for auxiliary sensor data (BMM150 does not have a FIFO). This is important for monitoring applications.

### 4.9.2 Interface Configuration

The configuration registers that control the auxiliary sensor interface operation, are only affecting the interface to the auxiliary sensor, not the configuration of the accelerometer sensor itself (this must be done in setup mode).

There are three basis configurations/modes of the auxiliary sensor interface:

- No auxiliary sensor access
- Setup mode: Auxiliary sensor access in manual mode
- Data mode: Auxiliary sensor access through hardware readout loop.

The setup of the auxiliary sensor itself must be done through the primary interface using indirect addressing in setup mode. When collecting sensor data, the BMA490L autonomously triggers the measurement of the auxiliary sensor using the auxiliary sensor forced mode and the data readout from the auxiliary sensor (data mode).

In setup mode, the auxiliary sensor may be configured and trim data may be read out from the auxiliary sensor. In the data mode the auxiliary sensor data are continuously copied into BMA490L registers and may be read out from BMA490L directly over the primary interface. For a BMM150 magnetometer, these are the auxiliary sensor data itself and Hall resistance, temperature is not required. The table below shows how to configure these three modes using the registers PWR_CONF, PWR_CTRL, and AUX_IF_CONF.aux_manual_en.

| Mode | AUX_IF_CONF.aux_manual_en | PWR_CONF.adv_power_save | PWR_CTRL.aux_en |
|---|---|---|---|
| No auxiliary sensor access | 1 | 1 | 0 |
| Setup mode | 1 | 0 | 0 |
| Data mode | 0 | x | 1 |

The auxiliary sensor interface mode may be enabled by setting bit IF_CONF.if_mode according to the following table.

| IF_CONF.if_mode | Result |
|---|---|
| 0 | Secondary IF disabled (default) |
| 1 | AuxIF enabled |

The auxiliary sensor interface operates at 400 kHz. This results in an I2C readout delay of about 250 us for 10 bytes of data.

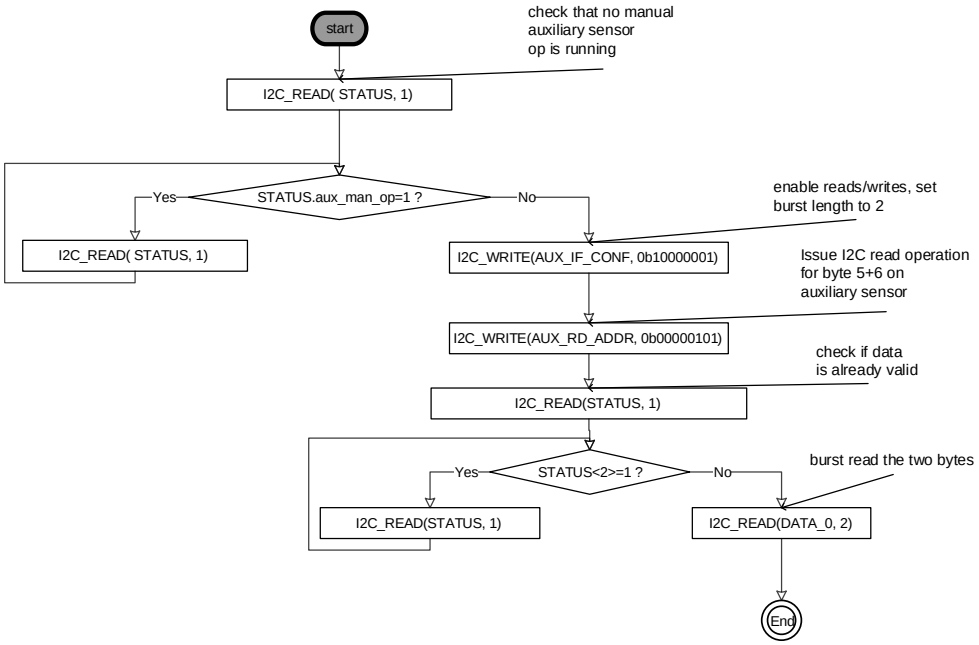
The I2C slave address of the auxiliary sensor is defined in AUX_DEV_ID. i2c_device_addr.

### 4.9.3 Setup mode (AUX_IF_CONF.aux_manual_en =0b1)

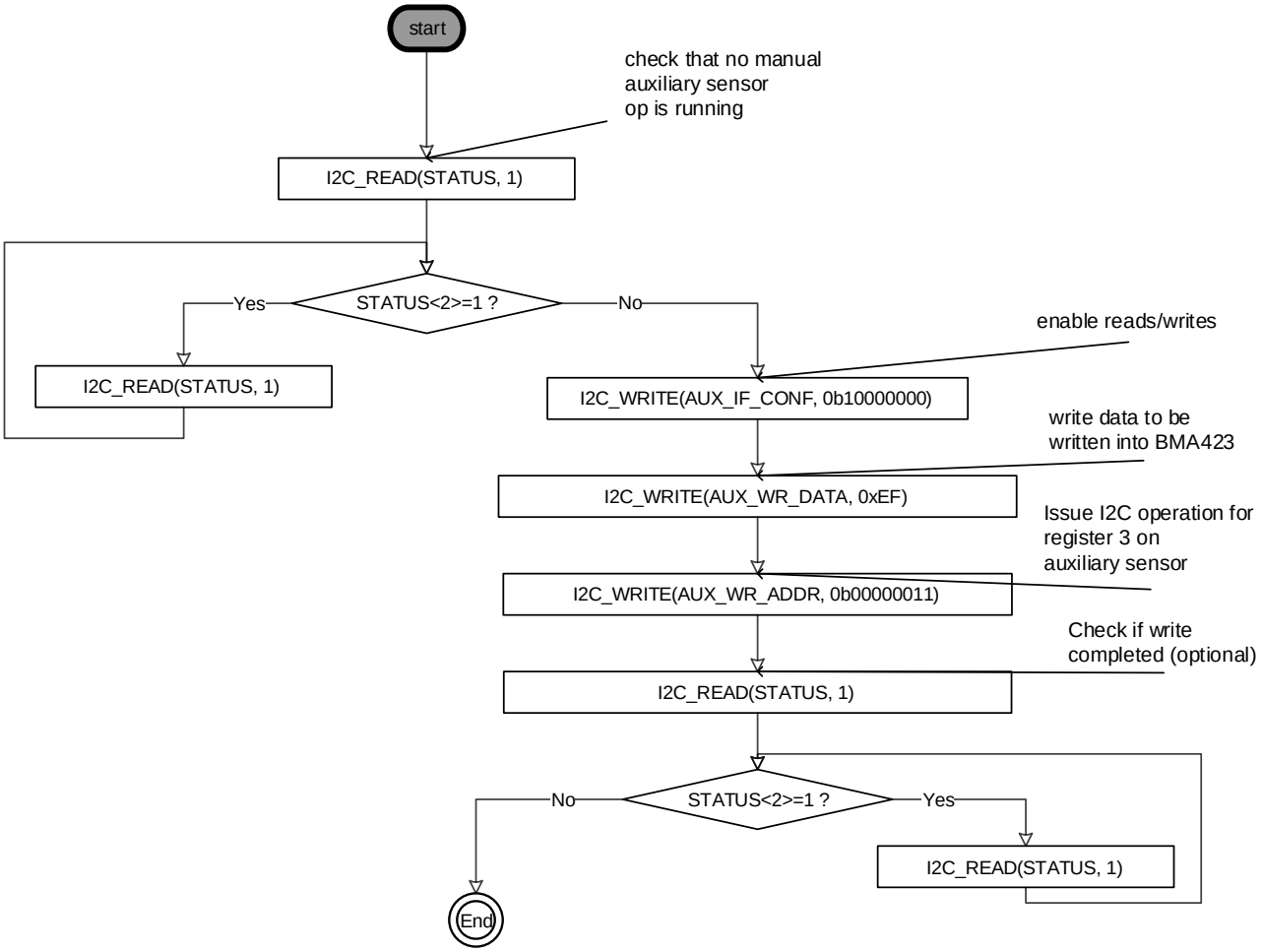
Through the primary interface the auxiliary sensor may be accessed using indirect addressing through the AUX_* registers. AUX_RD_ADDR and AUX_WR_ADDR define the address of the register to read/write in the auxiliary sensor register map and triggers the operation itself, when the auxiliary sensor interface is enabled through PWR_CTRL.aux_en.
For reads, the number of data bytes defined in AUX_IF_CONF.aux_rd_burst are read from the auxiliary sensor and written into the BMA490L Register DATA_0 to DATA_7. For writes only single bytes are written, independent of the settings in AUX_IF_CONF.aux_rd_burst. The data for the I2C write to auxiliary sensor must be stored in AUX_WR_DATA before the auxiliary sensor register address is written into AUX_WR_ADDR.
When a read or write operation is triggered by writing to AUX_RD_ADDR and AUX_WR_ADDR, STATUS.aux_man_op is set and it is reset when the operation is completed. For reads the DATA_0 to DATA_7 contains the read data, for writes AUX_WR_DATA may be overwritten again.

Configuration phase of the auxiliary sensor.

Example: Read bytes 5 and 6 of auxiliary sensor



Example: Write 0xEF into register 3 of auxiliary sensor



### 4.9.4 Data mode (AUX_IF_CONF.aux_manual_en=0)

AUX_RD_ADDR.read_addr defines the address of the data register from which to read the number of data bytes configured in AUX_IF_CONF.aux_rd_burst from AUX_0... AUX_7 data of the auxiliary sensor. These data are stored in the DATA_0 up to DATA_7 register. The data ready status is set in STATUS.drdy_aux, it is typically cleared through reading one of the DATA_0 to DATA_7 registers.
AUX_WR_ADDR.write_addr defines the register address of auxiliary sensor to start a measurement in forced mode in the auxiliary sensor register map. The delay (time offset) between triggering an auxiliary sensor measurement and reading the measurement data is specified in AUX_CONF.aux_offset. Reading of the data is done in a single I2C read operation with a burst length specified in AUX_IF_CONF.aux_rd_burst. For BMM150 AUX_IF_CONF.aux_rd_burst should be set to 0b11, i.e. 8 bytes. If AUX_IF_CONF.aux_rd_burst is set to a value lower than 8 bytes, the remaining auxiliary sensor data in the Register DATA_0 to DATA_7 and the FIFO are undefined.

It is recommended to disable the auxiliary sensor interface (IF_CONF.if_mode=0b0) before setting up AUX_RD_ADDR.read_addr and AUX_WR_ADDR.write_addr for the data mode. This does not put the auxiliary sensor itself into suspend mode but avoids gathering unwanted data during this phase. Afterwards the auxiliary sensor interface can be enabled (IF_CONF.if_mode=0b1) again.

### 4.9.5 Delay (Time Offset)

BMA490L supports starting the measurement of the sensor at the auxiliary sensor interface between 2.5 and 37.5 ms before the Register DATA are updated. This offset is defined in AUX_CONF.aux_offset. If set to 0b0, the measurement is done right after the last Register DATA update, therefore this measurement will be included in the next register DATA update.

## 4.10 **Sensor Self-Test**

The BMA490L has a comprehensive self test function for the MEMS element by applying electrostatic forces to the sensor core instead of external accelerations. By actually deflecting the seismic mass, the entire signal path of the sensor can be tested. Activating the self-test results in a static offset of the acceleration data; any external acceleration or gravitational force applied to the sensor during active self-test will be observed in the output as a superposition of both acceleration and self-test signal.
Before the self-test is enabled the g-range should be set to 8g. The self-test is activated for all axes by writing ACC_SELF_TEST.acc_self_test_en = 1b1. The self-test is disabled by writing ACC_SELF_TEST.acc_self_test_en = 1b0. It is possible to control the direction of the deflection through bit ACC_SELF_TEST.acc_self_test_sign. The excitation occurs in positive (negative) direction if ACC_SELF_TEST.acc_self_test_sign= 1b1 (’b0). The amplitude of the deflection has to be set low by writing ACC_SELF_TEST.acc_self_test_amp = 1b0. After the self-test is enabled, the user should wait 50ms before interpreting the acceleration data.
In order to ensure a proper interpretation of the self-test signal it is recommended to perform the self-test for both (positive and negative) directions and then to calculate the difference of the resulting acceleration values. The table below shows the minimum differences for each axis in order for the self test to pass. The actually measured signal differences can be significantly larger.

Self-test: Resulting minimum difference signal for BMA490L.

| | **x-axis signal** | **y-axis signal** | **z-axis signal** |
|---|---|---|---|
| BMA490L | 1800 mg | 1800 mg | 1800 mg |

It is recommended to perform a reset of the device after a self-test has been performed. If the reset cannot be performed, the following sequence must be kept to prevent unwanted interrupt generation: disable interrupts, change parameters of interrupts, wait for at least 50ms, and enable desired interrupts.

The recommended self test procedure is as follows:

1. Enable accelerometer with register PWR_CTRL.acc_en=1b1.
2. Set ±8g range in register ACC_RANGE.acc_range
3. Set self test amplitude to low by setting ACC_SELF_TEST.acc_self_test_amp = 1b0
4. Set ACC_CONF.acc_odr=1600Hz, Continuous sampling mode, ACC_CONF.acc_bwp=norm_avg4, ACC_CONF.acc_perf_mode=1b1.
5. Wait for > 2 ms
6. Enable self-test and set positive self-test polarity (ACC_SELF_TEST.acc_self_test_sign= 1b1)
7. Wait for > 50ms
8. Read and store positive acceleration value of each axis from registers DATA_8 to DATA_13
9. Enable self-test and set negative self-test polarity ACC_SELF_TEST.acc_self_test_sign= 1b0)
10. Wait for > 50ms
11. Read and store negative acceleration value of each axis from registers DATA_8 to DATA_13
12. Calculate difference of positive and negative acceleration values and compare against threshold values

## 4.11 Offset Compensation

BMA490L offers manual compensation as well as inline calibration.
Offset compensation is performed with pre-filtered data, and the offset is then applied to both, pre-filtered and filtered data. If necessary the result of this computation is saturated to prevent any overflow errors (the smallest or biggest possible value is set, depending on the sign).
The public offset compensation Registers OFFSET_0 to OFFSET_2 are images of the corresponding registers in the NVM. With each image update the contents of the NVM registers are written to the public registers. The public registers can be overwritten by the user at any time.
The offset compensation registers have a width of 8 bit using two's complement notation. The offset resolution (LSB) is 3.9 mg and the offset range is +- 0.5 g. Both are independent of the range setting.
Offset compensation needs to be enabled through NV_CONF.acc_off_en = 0b1

### 4.11.1 Manual Offset Compensation

The contents of the public compensation Register OFFSET_0 to OFFSET_2 may be set manually via the digital interface. After modifying the Register OFFSET_0 to OFFSET_2 the next data sample is not valid.
Offset compensation needs to be enabled through NV_CONF.acc_off_en.

### 4.11.2 Inline Calibration

For certain applications, it is often desirable to calibrate the offset once and to store the compensation values permanently. This can be achieved by using manual offset compensation to determine the proper compensation values and then storing these values permanently in the NVM.

Each time the device is reset, the compensation values are loaded from the non-volatile memory into the image registers and used for offset compensation.

## 4.12 **Non-Volatile Memory**

The registers NV_CONF and OFFSET_0 to OFFSET_2 have an NVM backup which are accessible by the user.
The content of the NVM is loaded to the image registers after a reset (either POR or softreset). As long as the image update is in progress, STATUS.cmd_rdy is 0b0, otherwise it is 0b1.
The image registers can be read and written like any other register.
Writing to the NVM is a 4-step procedure:

1. Set PWR_CONF.adv_power_save = 0b0
2. Write the new contents to the image registers.
3. Write 0b1 to bit NVM_CONF.nvm_prog_en in order to unlock the NVM.
4. Write nvm_prog to the CMD register to trigger the write process.
5. Write 0b0 to bit NVM_CONF.nvm_prog_en in order to lock the NVM, after the write process is completed

Writing to the NVM always renews the entire NVM contents. It is possible to check the write status by reading *STATUS.cmd_rdy*. While *STATUS.cmd_rdy* = 0b0, the write process is still in progress; when *STATUS.cmd_rdy* = 0b1, writing is completed. An NVM write cycle can only be initiated, if PWR_CONF.adv_power_save = 0b0.
Until boot phase is finished (after POR or softreset), the serial interface is not operational. The NVM shadow registers must not be accessed during an ongoing NVM command (initiated through the Register CMD). In all other cases, register can be read or written.
As long as an NVM read (during sensor boot and soft reset) or an NVM write is ongoing, writes to sensor registers are discarded, reads return the Register STATUS independent of the read address.

## 4.13 **Soft-Reset**

A softreset can be initiated at any time by writing the command *softreset (0xB6)* to register CMD. The softreset performs a fundamental reset to the device which is largely equivalent to a power cycle. Following a delay, all user configuration settings are overwritten with their default state (setting stored in the NVM) wherever applicable. This command is functional in all operation modes but must not be performed while NVM writing operation is in progress.

# 5. Register Description

## 5.1 General Remarks

Registers can be read and written in all power configurations with the exception of FEATURES_IN and FIFO_DATA which need PWR_CONF.adv_power_save set to 0b0. The following chapter contains only the general register map, feature related registers are excluded.

## 5.2 Register Map

| read/write | read only | write only | reserved |
|---|---|---|---|

| Register Address | Register Name | Default Value | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | ID: |
| 0x7E | CMD | 0x00 | cmd | | | | | | | |
| 0x7D | PWR_CTRL | 0x00 | reserved | | | | | acc_en | reserved | aux_en |
| 0x7C | PWR_CONF | 0x03 | reserved | | | | | | fifo_self_wakeup | adv_power_save |
| 0x7B | - | - | reserved | | | | | | | |
| ... | - | - | reserved | | | | | | | |
| 0x74 | - | - | reserved | | | | | | | |
| 0x73 | OFFSET_2 | 0x00 | off_acc_z | | | | | | | |
| 0x72 | OFFSET_1 | 0x00 | off_acc_y | | | | | | | |
| 0x71 | OFFSET_0 | 0x00 | off_acc_x | | | | | | | |
| 0x70 | NV_CONF | 0x00 | reserved | | | | acc_off_en | i2c_wdt_en | i2c_wdt_sel | spi_en |
| 0x6F | - | - | reserved | | | | | | | |
| 0x6E | - | - | reserved | | | | | | | |
| 0x6D | ACC_SELF_TEST | 0x00 | reserved | | | | acc_self_test_amp | acc_self_test_sign | reserved | acc_self_test_en |
| 0x6C | - | - | reserved | | | | | | | |
| 0x6B | IF_CONF | 0x00 | reserved | | | if_mode | reserved | | | spi3 |
| 0x6A | NVM_CONF | 0x00 | reserved | | | | | | nvm_prog_en | reserved |
| 0x69 | - | - | reserved | | | | | | | |
| ... | - | - | reserved | | | | | | | |
| 0x60 | - | - | reserved | | | | | | | |
| 0x5F | INTERNAL_ERROR | 0x00 | reserved | | | | | int_err_2 | int_err_1 | reserved |
| 0x5E | FEATURES_IN | 0x00 | features_in | | | | | | | |
| 0x5D | - | - | reserved | | | | | | | |
| ... | - | - | reserved | | | | | | | |
| 0x5A | - | - | reserved | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0x59 | INIT_CTRL | 0x90 | init_ctrl | | | | | | |
| 0x58 | INT_MAP_DATA | 0x00 | reserved | int2_drdy | int2_fwm | int2_ffull | reserved | int1_drdy | int1_fwm | int1_ffull |
| 0x57 | INT2_MAP | 0x00 | error_int_out | no_motion_out | any_motion_out | reserved | | | | |
| 0x56 | INT1_MAP | 0x00 | error_int_out | no_motion_out | any_motion_out | reserved | | | | |
| 0x55 | INT_LATCH | 0x00 | reserved | | | | | | | int_latch |
| 0x54 | INT2_IO_CTRL | 0x00 | reserved | | | | input_en | output_en | od | lvl | edge_ctrl |
| 0x53 | INT1_IO_CTRL | 0x00 | reserved | | | | input_en | output_en | od | lvl | edge_ctrl |
| 0x52 | - | - | reserved | | | | | | | |
| ... | - | - | reserved | | | | | | | |
| 0x50 | - | - | reserved | | | | | | | |
| 0x4F | AUX_WR_DATA | 0x02 | write_data | | | | | | | |
| 0x4E | AUX_WR_ADDR | 0x4C | write_addr | | | | | | | |
| 0x4D | AUX_RD_ADDR | 0x42 | read_addr | | | | | | | |
| 0x4C | AUX_IF_CONF | 0x83 | aux_manual_en | reserved | | | | | aux_rd_burst | |
| 0x4B | AUX_DEV_ID | 0x20 | i2c_device_addr | | | | | | | reserved |
| 0x4A | - | - | reserved | | | | | | | |
| 0x49 | FIFO_CONFIG_1 | 0x10 | reserved | fifo_acc_en | fifo_aux_en | fifo_header_en | fifo_tag_int1_en | fifo_tag_int2_en | reserved | |
| 0x48 | FIFO_CONFIG_0 | 0x02 | reserved | | | | | | fifo_time_en | fifo_stop_on_full |
| 0x47 | FIFO_WTM_1 | 0x02 | reserved | | | | fifo_water_mark_12_8 | | | |
| 0x46 | FIFO_WTM_0 | 0x00 | fifo_water_mark_7_0 | | | | | | | |
| 0x45 | FIFO_DOWNS | 0x80 | acc_fifo_filt_data | acc_fifo_downs | | | reserved | | | |
| 0x44 | AUX_CONF | 0x46 | aux_offset | | | | aux_odr | | | |
| 0x43 | - | - | reserved | | | | | | | |
| 0x42 | - | - | reserved | | | | | | | |
| 0x41 | ACC_RANGE | 0x01 | reserved | | | | | | acc_range | |
| 0x40 | ACC_CONF | 0xA8 | acc_perf_mode | acc_bwp | | | acc_odr | | | |
| 0x3F | - | - | reserved | | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ... | - | - | reserved | | | | | |
| 0x2B | - | - | reserved | | | | | |
| 0x2A | INTERNAL_STATUS | 0x00 | reserved | odr_50Hz_error | axes_remap_error | message | | |
| 0x29 | - | - | reserved | | | | | |
| ... | - | - | reserved | | | | | |
| 0x27 | - | - | reserved | | | | | |
| 0x26 | FIFO_DATA | 0x00 | fifo_data | | | | | |
| 0x25 | FIFO_LENGTH_1 | 0x00 | reserved | fifo_byte_counter_13_8 | | | | |
| 0x24 | FIFO_LENGTH_0 | 0x00 | fifo_byte_counter_7_0 | | | | | |
| 0x23 | - | - | reserved | | | | | |
| 0x22 | TEMPERATURE | 0x00 | temperature | | | | | |
| 0x21 | - | - | reserved | | | | | |
| ... | - | - | reserved | | | | | |
| 0x1E | - | - | reserved | | | | | |
| 0x1D | INT_STATUS_1 | 0x00 | acc_drdy_int | reserved | aux_drdy_int | reserved | fwm_int | ffull_int |
| 0x1C | INT_STATUS_0 | 0x00 | error_int_out | no_motion_out | any_motion_out | reserved | | |
| 0x1B | EVENT | 0x01 | reserved | | | | | por_detected |
| 0x1A | SENSORTIME_2 | 0x00 | sensor_time_23_16 | | | | | |
| 0x19 | SENSORTIME_1 | 0x00 | sensor_time_15_8 | | | | | |
| 0x18 | SENSORTIME_0 | 0x00 | sensor_time_7_0 | | | | | |
| 0x17 | DATA_13 | 0x00 | acc_z_15_8 | | | | | |
| 0x16 | DATA_12 | 0x00 | acc_z_7_0 | | | | | |
| 0x15 | DATA_11 | 0x00 | acc_y_15_8 | | | | | |
| 0x14 | DATA_10 | 0x00 | acc_y_7_0 | | | | | |
| 0x13 | DATA_9 | 0x00 | acc_x_15_8 | | | | | |
| 0x12 | DATA_8 | 0x00 | acc_x_7_0 | | | | | |
| 0x11 | DATA_7 | 0x00 | aux_r_15_8 | | | | | |
| 0x10 | DATA_6 | 0x00 | aux_r_7_0 | | | | | |
| 0x0F | DATA_5 | 0x00 | aux_z_15_8 | | | | | |
| 0x0E | DATA_4 | 0x00 | aux_z_7_0 | | | | | |
| 0x0D | DATA_3 | 0x00 | aux_y_15_8 | | | | | |
| 0x0C | DATA_2 | 0x00 | aux_y_7_0 | | | | | |
| 0x0B | DATA_1 | 0x00 | aux_x_15_8 | | | | | |
| 0x0A | DATA_0 | 0x00 | aux_x_7_0 | | | | | |
| 0x09 | - | - | reserved | | | | | |
| ... | - | - | reserved | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04 | - | - | reserved | | | | | | | |
| 0x03 | STATUS | 0x10 | drdy_acc | reserved | drdy_aux | cmd_rdy | reserved | aux_man_op | reserved | |
| 0x02 | ERR_REG | 0x00 | aux_err | fifo_err | reserved | error_code | | | cmd_err | fatal_err |
| 0x01 | - | - | reserved | | | | | | | |
| 0x00 | CHIP_ID | 0x1A | chip_id | | | | | | | |

FEATURES_IN

| Register Address | Register Name | Default Value | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x5E: 0x0B | general_settings.axes_remapping[1] | 0x00 | reserved | | | | | | | map_z_axis_sign |
| 0x5E: 0x0A | general_settings.axes_remapping[0] | 0x88 | map_z_axis | | map_y_axis_sign | | map_y_axis | | map_x_axis_sign | map_x_axis |
| 0x5E: 0x09 | general_settings.Reserved[1] | 0x00 | Reserved | | | | | | | |
| 0x5E: 0x08 | general_settings.Reserved[0] | 0x00 | Reserved | | | | | | | |
| 0x5E: 0x07 | no_motion.settings_2[1] | 0x00 | z_en | y_en | x_en | duration | | | | |
| 0x5E: 0x06 | no_motion.settings_2[0] | 0x05 | duration | | | | | | | |
| 0x5E: 0x05 | no_motion.settings_1[1] | 0x00 | reserved | | | | | | threshold | |
| 0x5E: 0x04 | no_motion.settings_1[0] | 0xAA | threshold | | | | | | | |
| 0x5E: 0x03 | any_motion.settings_2[1] | 0x00 | z_en | y_en | x_en | duration | | | | |
| 0x5E: 0x02 | any_motion.settings_2[0] | 0x05 | duration | | | | | | | |
| 0x5E: 0x01 | any_motion.settings_1[1] | 0x00 | reserved | | | | | | threshold | |

| | | | |
|---|---|---|---|
| 0x5E: 0x00 | any_motion.settings_1[0] | 0xAA | threshold |

### 5.2.1 Register (0x00) CHIP_ID

DESCRIPTION: Chip identification code
RESET: 0x1A
DEFINITION (Go to register map):

| Name | Register (0x00) CHIP_ID | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 1 |
| Content | chip_id | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 1 | 0 | 1 | 0 |
| Content | chip_id | | | |

chip_id: Chip identification code for BMA490L

### 5.2.2 Register (0x02) ERR_REG

DESCRIPTION: Reports sensor error conditions
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x02) ERR_REG | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | n/a | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_err | fifo_err | reserved | error_code |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | error_code | | cmd_err | fatal_err |

fatal_err: Fatal Error, chip is not in operational state (Boot-, power-system). This flag will be reset only by power-on-reset or softreset.
cmd_err: Command execution failed.
error_code: Error codes for persistent errors

| error_code | | |
|---|---|---|
| 0x00 | no_error | no error is reported |
| 0x01 | acc_err | error in Register ACC_CONF |

fifo_err: Error in FIFO detected: Input data was discarded in stream mode. This flag will be reset when read.
aux_err: Error in I2C-Master detected. This flag will be reset when read.

### 5.2.3 Register (0x03) STATUS

DESCRIPTION: Sensor status flags
RESET: 0x10
DEFINITION (Go to register map):

| Name | Register (0x03) STATUS | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | n/a | R | R |
| Reset Value | 0 | 0 | 0 | 1 |
| Content | drdy_acc | reserved | drdy_aux | cmd_rdy |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | n/a | R | n/a | n/a |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | reserved | aux_man_op | reserved | |

aux_man_op: '1'('0') indicate a (no) manual auxiliary interface operation is ongoing.
cmd_rdy: CMD decoder status. `0´ -> Command in progress `1´ -> Command decoder is ready to accept a new command
drdy_aux: Data ready for auxiliary sensor. It gets reset when one auxiliary DATA register is read out
drdy_acc: Data ready for accelerometer. It gets reset when one accelerometer DATA register is read out

### 5.2.4 Register (0x0A) DATA_0

DESCRIPTION: AUX_X(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0A) DATA_0 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_x_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_x_7_0 | | | |

### 5.2.5 Register (0x0B) DATA_1

DESCRIPTION: AUX_X(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0B) DATA_1 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_x_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_x_15_8 | | | |

### 5.2.6 Register (0x0C) DATA_2

DESCRIPTION: AUX_Y(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0C) DATA_2 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_y_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_y_7_0 | | | |

### 5.2.7 Register (0x0D) DATA_3

DESCRIPTION: AUX_Y(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0D) DATA_3 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_y_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_y_15_8 | | | |

### 5.2.8 Register (0x0E) DATA_4

DESCRIPTION: AUX_Z(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0E) DATA_4 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_z_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_z_7_0 | | | |

### 5.2.9 Register (0x0F) DATA_5

DESCRIPTION: AUX_Z(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x0F) DATA_5 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_z_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_z_15_8 | | | |

### 5.2.10 Register (0x10) DATA_6

DESCRIPTION: AUX_R(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x10) DATA_6 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_r_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_r_7_0 | | | |

### 5.2.11 Register (0x11) DATA_7

DESCRIPTION: AUX_R(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x11) DATA_7 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_r_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | aux_r_15_8 | | | |

### 5.2.12 Register (0x12) DATA_8

DESCRIPTION: ACC_X(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x12) DATA_8 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_x_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_x_7_0 | | | |

### 5.2.13 Register (0x13) DATA_9

DESCRIPTION: ACC_X(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x13) DATA_9 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_x_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_x_15_8 | | | |

### 5.2.14 Register (0x14) DATA_10

DESCRIPTION: ACC_Y(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x14) DATA_10 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_y_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_y_7_0 | | | |

### 5.2.15 Register (0x15) DATA_11

DESCRIPTION: ACC_Y(MSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x15) DATA_11 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_y_15_8 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_y_15_8 | | | |

### 5.2.16 Register (0x16) DATA_12

DESCRIPTION: ACC_Z(LSB)
RESET: 0x00
DEFINITION (Go to register map):

| Name | Register (0x16) DATA_12 | | | |
|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_z_7_0 | | | |
| Bit | 3 | 2 | 1 | 0 |
| Read/Write | R | R | R | R |
| Reset Value | 0 | 0 | 0 | 0 |
| Content | acc_z_7_0 | | | |