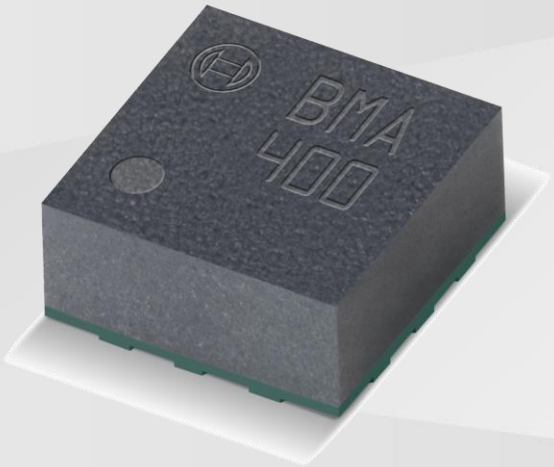
BMA400

3-axes ultra-low power accelerometer



BMA400 – Data sheet

Document revision	1.6
Document release date	April 2021
Document number	BST-BMA400-DS000-07
Technical reference code	0 273 141 275
Notes	Data and descriptions in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product appearance

BMA400 – Basic Description

12 bit, digital, triaxial acceleration sensor with smart on-chip motion and position-triggered interrupt features.

Key features

- Small package size
  LGA package (12 pins), footprint 2mm x 2mm, height 0.95 mm
- Ultra-low power
  Low current consumption of data acquisition without compromising on performance (< 14.5 µA with highest performance)
- Programmable functionality
  Acceleration ranges ±2g/±4g/±8g/±16g
  Low-pass filter bandwidths = 0.48*ODR
  up to a max. output data read out of 800Hz
- On-chip FIFO
  Integrated FIFO on sensor with 1 KB
- On-chip interrupt features
  Auto-low power/Auto wakeup
  Activity/In-activity
  Step Counter (overall device current consumption 4µA)
  Activity Recognition (Walking, Running, Standing still)
  Orientation detection
  Tap/double tap
- Digital interface
  SPI (4-wire, 3-wire), I²C, 2 interrupt pins
  V_{DDIO} voltage range: 1.2V to 3.6V
- RoHS compliant, halogen-free

Typical applications

- Step Counting with ultra-low current consumption for extensive battery lifetime
- Advanced system power management for mobile applications and (smart) watches
- Fitness applications / Activity Tracking
- Tap / double tap sensing
- Drop detection for warranty logging
- Window/door measurements for climate control and alarm systems
- IoT applications powered by coin cell driven batteries, requiring <1uA and auto-wakeup functionality

Index of Contents

1. Specification

Unless stated otherwise, the given values are over lifetime, operating temperature and voltage ranges. Minimum/maximum values are ±3σ.

Parameter Specification

Parameter	Symbol	Condition	Min	Typ	Max	Units
Acceleration Range	g_{FS2g}			±2		g
	g_{FS4g}			±4		g
	g_{FS8g}			±8		g
	g_{FS16g}			±16		g
Supply Voltage Internal Domains	V_{DD}		1.72	1.8	3.6	V
Supply Voltage I/O Domain	V_{DDIO}		1.2	1.8	3.6	V
Voltage Input Low Level	V_{IL}	SPI & I²C			$0.3V_{DDIO}$	
Voltage Input High Level	V_{IH}	SPI & I²C	$0.7V_{DDIO}$			
Voltage Output Low Level	V_{OL}	V_{DDIO}=1.8V, I_{OL}=3mA, SPI			$0.2V_{DDIO}$	
		V_{DDIO}=1.2V, I_{OL}=3mA, SPI			$0.23V_{DDIO}$	
Voltage Output High Level	V_{OH}	V_{DDIO}=1.8V, I_{OH}=3mA, SPI	$0.8V_{DDIO}$			
		V_{DDIO}=1.2V, I_{OH}=3mA, SPI	$0.62V_{DDIO}$			
Total Supply Current in Normal mode	I_{DD}	Nominal VDD and VDDIO, 25°C, OSR=3		14.5		µA
		OSR=0		3.5		µA
Total Supply Current in Sleep Mode	I_{DDsum}	Nominal VDD and VDDIO, 25°C		160		nA
Total Supply Current in Low-power Mode	I_{DDlp1}	Nominal VDD and VDDIO, 25°C 25 Hz ODR OSR=0		850		nA
Wake-Up Time	t_{w_up}	From sleep to normal mode			2/ODR	
Power-Up Time	t_{s_up}	Starting the device to sleep mode			1	ms
Operating Temperature	T_A		-40		+85	°C

OUTPUT SIGNAL						
Parameter	**Symbol**	**Condition**	**Min**	**Typ**	**Max**	**Units**
Sensitivity	S_{2g}	g_{FS2g}, T_A=25°C		1024		LSB/g
	S_{4g}	g_{FS4g}, T_A=25°C		512		LSB/g
	S_{8g}	g_{FS8g}, T_A=25°C		256		LSB/g
	S_{16g}	g_{FS16g}, T_A=25°C		128		LSB/g
Sensitivity Temperature Drift	TCS	Nominal V_{DD} and VDD_{IO}, g_{FS4g} Over life-time		0.025		%/K
Zero-g Offset	Off	Nominal V_{DD} and VDD_{IO}, 25°C, g_{FS4g} Over life-time		50		mg
Zero-g Offset Temperature Drift	TCO	Nominal V_{DD} and VDD_{IO}, g_{FS4g} Over life-time		1		mg/K
Output Data Rate	ODR_{NORM}	Normal mode	12.5		800	Hz
	ODR_{LPM}	Low-power mode		25		Hz
Bandwidth	BW_{norm}	3dB cutoff frequency is selectable in normal mode	0.24 x ODR_{NORM}		0.48 x ODR_{NORM}	Hz
Nonlinearity	NL	Nominal V_{DD} and VDD_{IO}, 25°C, g_{FS4g}		0.5		%FS
Output Noise Density	n_{rms}	Typical V_{DD} and VDD_{IO}, normal mode, OSR=3 (high performance) 25°C, 4g (X,Y-Axis)		180		µg/√Hz
		Typical V_{DD} and VDD_{IO}, normal mode, OSR=3 (high performance) 25°C, 4g (Z-Axis)		240		µg/√Hz

MECHANICAL CHARACTERISTICS						
Parameter	**Symbol**	**Condition**	**Min**	**Typ**	**Max**	**Units**
Cross Axis Sensitivity	S	relative contribution between any two of the three axes		2		%
Alignment Error	E_A	relative to package outline		0.5		°

2. Absolute maximum ratings

Absolute maximum ratings

Parameter	Condition	Min	Max	Units
Voltage at Supply Pin	V_{DD} Pin	-0.3	4	V
	V_{DDIO} Pin	-0.3	4	V
Voltage at any Logic Pin	Non-Supply Pin	-0.3	V_{DDIO}+0.3, <4	V
Passive Storage Temp. Range	≤ 65% rel. H.	-50	+150	°C
Mechanical Shock	Duration ≤ 200µs		10,000	g
	Duration ≤ 1.0ms		2,000	g
	Free fall onto hard surfaces		1.8	m
ESD	HBM, at any Pin		2	kV
	CDM		500	V
	MM		200	V

Note:
Stress above these limits may cause damage to the device. Exceeding the specified electrical limits may affect the device reliability or cause malfunction.

3. Quick Start Guide

The purpose of this chapter is to help developers who want to start working with the BMA400 by giving you some very basic hands-on application examples to get started.

Note about using the BMA400:

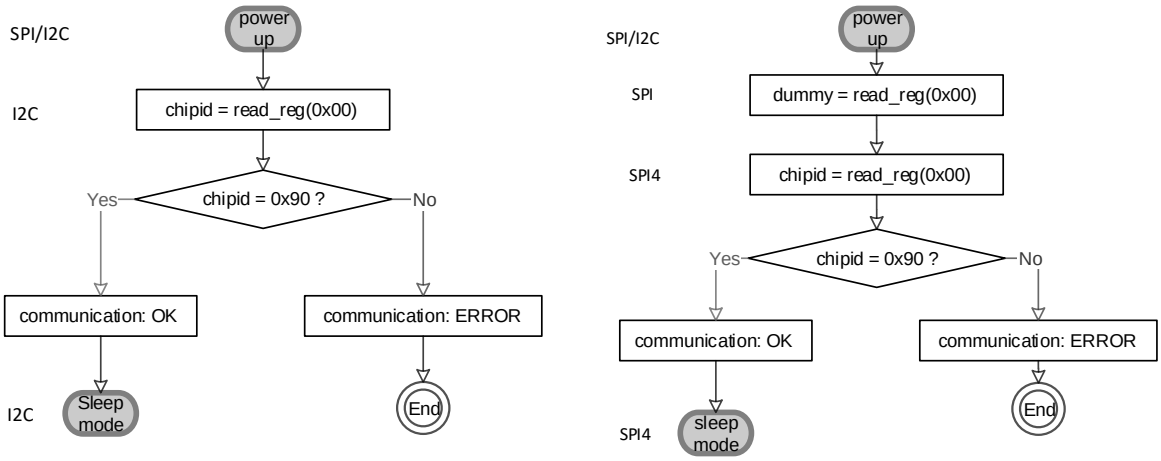
- The communication between application processor AP and BMA400 will happen either over I2C or SPI interface. For more information about the interfaces, read the related chapter 6. Digital Interfaces.
- For information about connecting the BMA400 to the host (AP), read the related chapter 7. It describes Pin-out and Connection Diagrams.

First application setup examples algorithms:

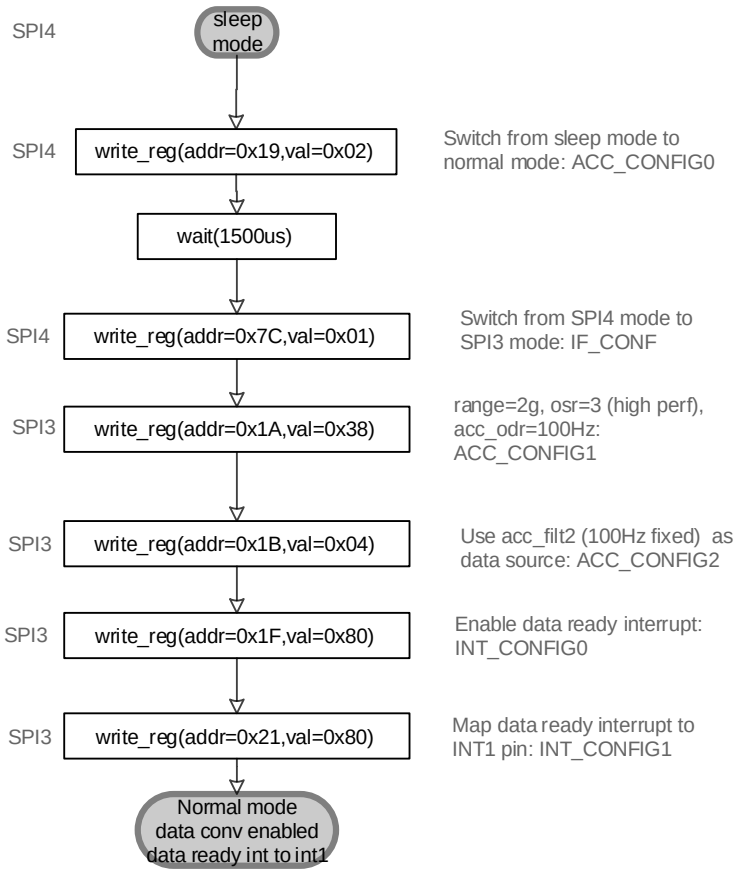
After correct power up by setting the correct voltage to the power pins, the BMA400 enters automatically into the Power On Reset (POR) sequence, also called boot sequence. After having completed boot, the BMA400 enters sleep mode where it consumes 160nA. No data conversions happen in this phase, but register read-out and write is possible. Communication can start in I2C or SPI mode. The BMA400 automatically detects which format is used. When SPI format is used, the BMA400 switches to SPI4 mode and remains in this mode until reset. The switching to SPI requires to send the very first SPI packet twice: the first packet will be ignored by the BMA400.
If SPI3 communication is desired, a write to register IF_CONF (write_reg(IF_CONF, 0x01)) switches the communication protocol to SPI3.
In order to properly make use of the BMA400, certain steps from host processor side are needed. The most typical operations will be explained in the following application examples in form of flow-diagrams.

1. **Example 1:** ***Testing communication with the BMA400, switch to SPI communication, state data conversion, enable data ready interrupt and map it to INT1 pin***

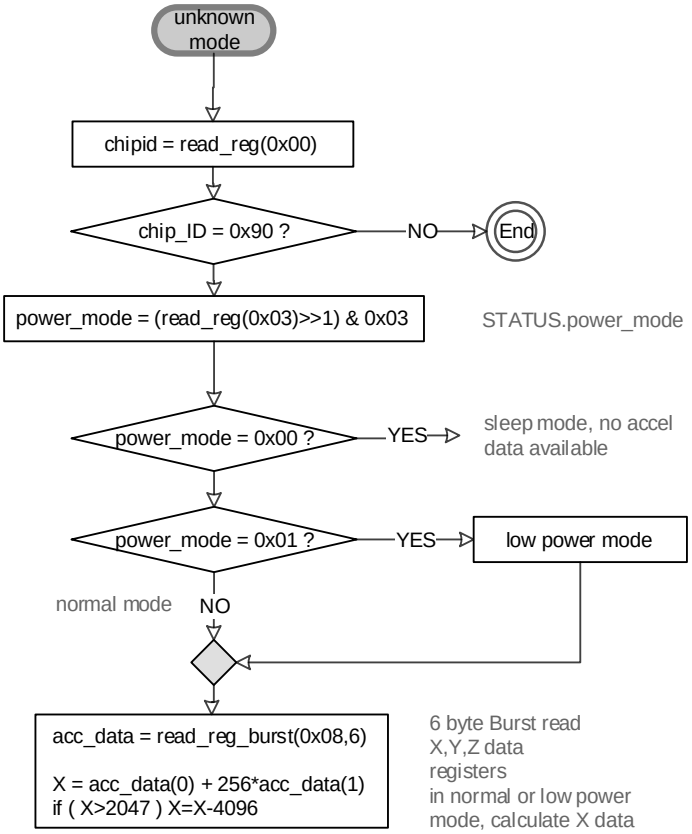
-reading chip id (checking correct communication) using I2C or SPI



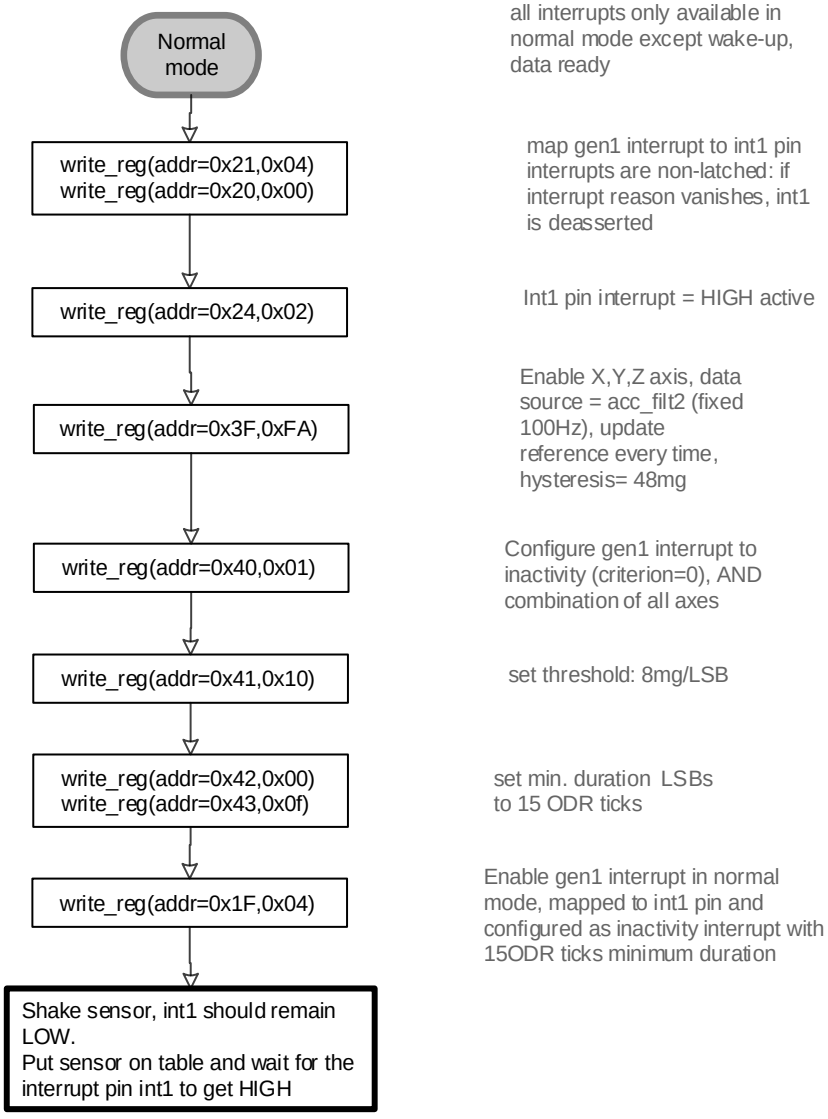
–switching from sleep to normal mode, then SPI3 mode, then enable data ready interrupt and map to pin int1



-checking communication via chipid, check power mode, read acceleration data if not in sleep mode



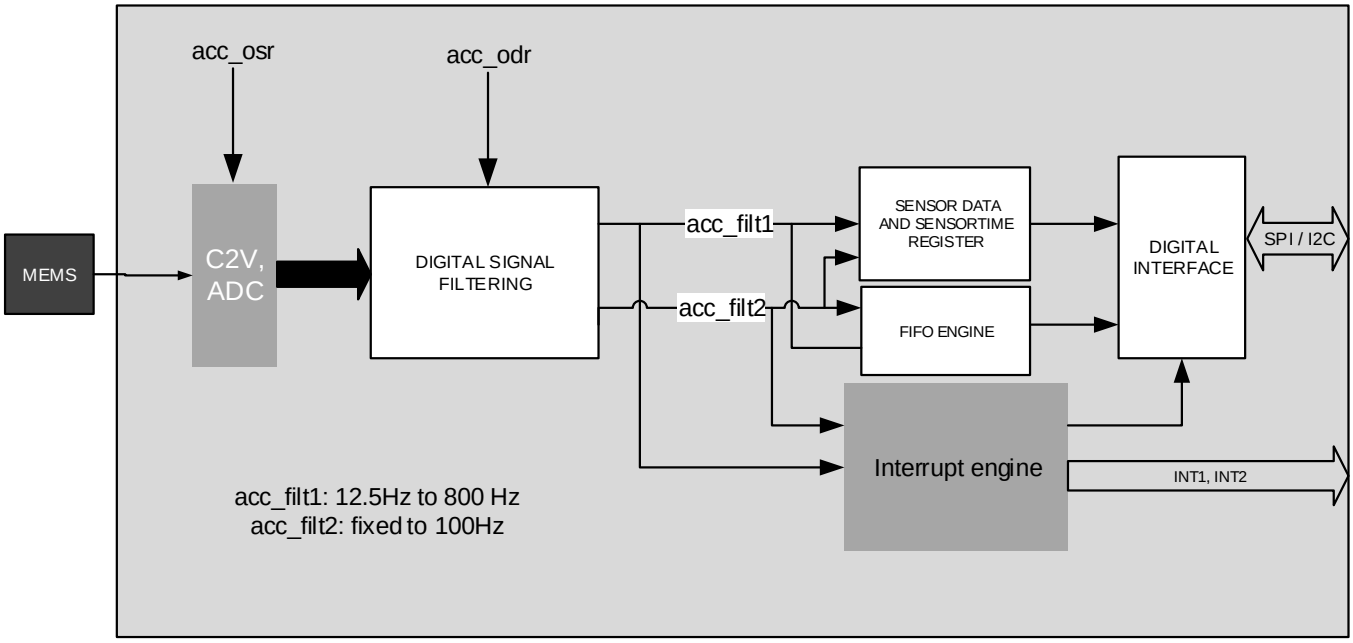
1. **Example 3:** ***Testing interrupt engine of BMA400 (example: inactivity interrupt)***
   a. -performing reconfiguration sequence (interrupt feature: significant motion)



Further steps:
The BMA400 has many more capabilities that are described in this document and include FIFO, power saving modes, synchronization capabilities with host processor, data synchronization, many interrupts generation and more features like step counter, etc.

4. Functional Description

4.1. Block Diagram



4.2. Supply Voltage and Power Management

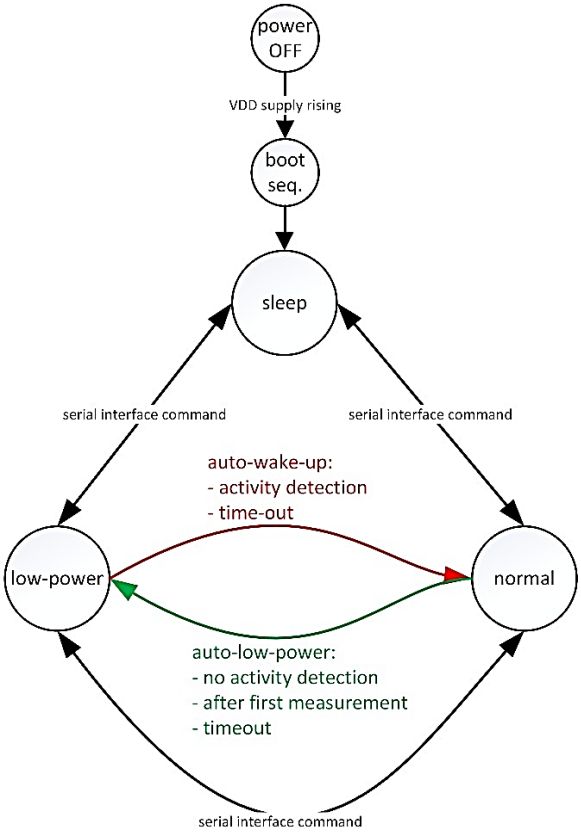
BMA400 has two distinct power supply pins:

- VDD is the main power supply.
- VDDIO is a separate power supply pin used for supplying power for the digital communication interface.

There are no limitations with respect to the voltage level applied to the VDD and VDDIO pins, as long as it lies within the respective operating range. Furthermore, the device can be completely switched off (VDD=0V) while keeping the VDDIO supply within operating range or vice versa. However if the VDDIO supply is switched off, all interface pins (CSB, SDX, SCX) must be kept close to GNDIO potential. No constraints exist for the minimum slew-rate of the voltage applied to the VDD and VDDIO pins.

4.3. Power Modes – performance modes

The power mode and all major settings affecting performance, current consumption, noise and output data rate are controlled in registers ACC_CONFIG0, ACC_CONFIG1 and ACC_CONFIG2.
The BMA400 knows three power modes: sleep mode, low-power mode and normal mode.



In **sleep mode**, current consumption is 200nA, and data conversions are stopped as well as sensortime functionality.

In **low power mode**, data conversion runs with a fixed rate of 25Hz, and performance can be controlled via ACC_CONFIG0.osr_lp setting. Current consumption ranges between 800 nA and 1200 nA depending on performance setting. The low power mode should be mainly used in combination with activity detection as self wake-up mode. In this use case, 800 nA are sufficient.

In **normal mode**, output data rates between 800Hz and 12.5Hz can be configured using the registers ACC_CONFIG1.acc_odr and ACC_CONFIG1.osr. The noise density performance of the BMA400 is mainly determined by ACC_CONFIG1.osr. The RMS noise and the resulting current consumption of the device is influenced by ACC_CONFIG1.acc_odr and ACC_CONFIG1.osr.

In all three power modes both register contents and FIFO contents are retained. FIFO readout and flushing can be done in normal and low power mode. The FIFO is written only in normal mode.

ACC_CONFIG0. power_mode<1:0>	Description	Details

b00 b11	Sleep mode (default state after power-up and after reset)	No sensortime. No FIFO read, no data conversions. Register and FIFO content retained, registers readable and writeable.
b01	Low-power mode	Data conversion at 25Hz fixed. Noise performance and current consumption tunable by ACC_CONFIG0.osr_lp setting. Auto wake-up function using wake-up interrupt or timer to switch automatically into normal mode. FIFO read is supported. No FIFO write.
b10	Normal mode	Data conversion configurable between 800Hz and 12.5Hz. Noise performance and current consumption tunable by ACC_CONFIG1.osr. FIFO read and write All interrupts available Auto-low-power function using generic interrupt 1 or timer to switch automatically into low-power mode.

Current consumption (uA) in normal mode and low-power mode

	ACC_CONFIG1.osr / ACC_CONFIG0.osr_lp			
	11	**10**	**01**	**00**
Normal mode ACC_CONFIG0. power_mode<1:0> = b10	14.5	9.5	5.8	3.5
Low-power mode ACC_CONFIG0. power_mode<1:0> = b01	1.35	1.1	0.93	0.85

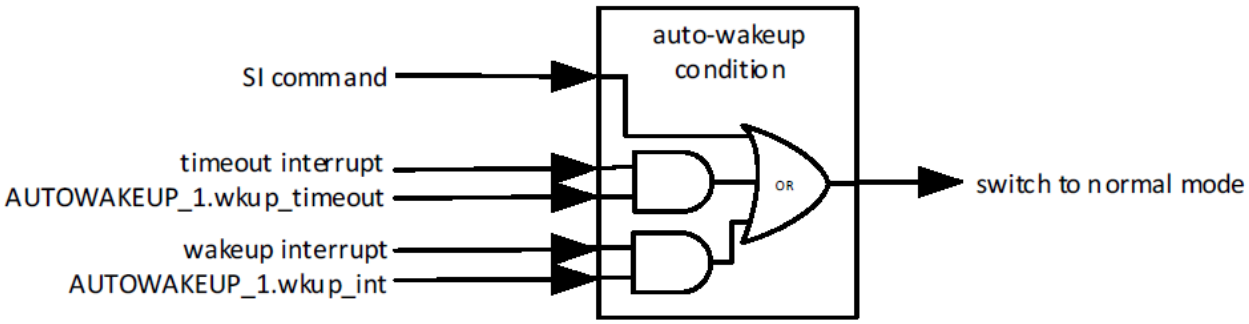
Noise performace (rms in mg) in normal mode and low-power mode in 4g range (x and y axes are shown, Z-axis is 1.45 x higher)

		ACC_CONFIG1.osr / ACC_CONFIG0.osr_lp			
	ODR [Hz]	**11**	**10**	**01**	**00**
Normal mode ACC_CONFIG0. power_mode<1:0> = b10	**800**	4.23	5.49	7.39	10.24
	400	2.98	3.89	5.23	7.24
	200	2.11	2.74	3.70	5.13
	100	1.50	1.93	2.61	3.63
	50	1.05	1.38	1.84	2.55
	25	0.74	0.97	1.31	1.81
	12.5	0.54	0.68	0.92	1.29
Low-power mode ACC_CONFIG0. power_mode<1:0> = b01	**25**	3.92	5.53	7.75	10.96

Auto wake-up

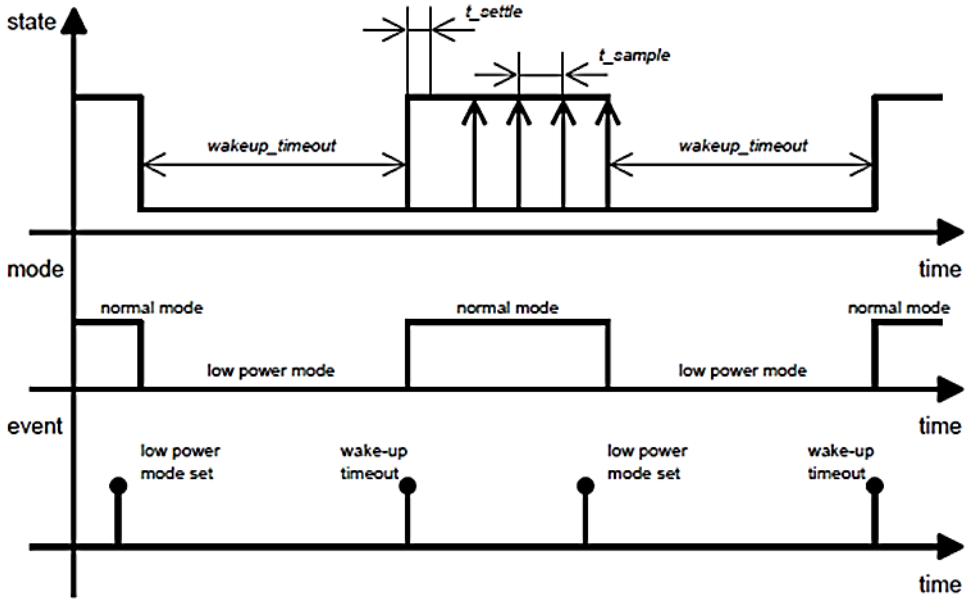
The auto-wakeup function is part of the power management concept of the BMA400. If the wakeup function (only available in low-power mode) changes the power mode to “normal”, the host processor can be notified by an interrupt. This is called “wakeup interrupt”, thus, the two topics “auto wakeup” and “wakeup interrupt” are handled together in this chapter.
The transition from Low-power to Normal mode is named “wake-up”.
Switching into Normal mode from Low-power mode can be explicitly triggered by a serial interface command. This can also be done automatically by using the auto wakeup function.
Auto wakeup can be either timer triggered or activity triggered. Each selected condition is independent and can be used as wake-up condition. In case more than one condition is selected, the first occurred condition sets the BMA400 into normal mode.
The three possible triggers for wake-up from low-power mode are:

- by serial command
- by timeout
- by wake-up interrupt on activity



Wakeup by timeout
The source condition wkup_timeout (enabling bit) and the timeout counter threshold value AUTOWAKEUP(0/1).wakeup_timeout_thres is configured in register AUTOWAKEUP_(0/1).

The wakeup_timeout_thres has 12bits for configuration of counter duration, with a resolution of 2.5ms/LSB. The maximum timeout for wake-up is 10.24s (4096*2.5ms).



Wake-up interrupt on activity

In low-power mode BMA400 wake-up interrupt is triggered, if the conditions as defined by the configuration registers are fulfilled. This function is always on in low power mode, and can be mapped to interrupt pin. The wake-up can be used for auto wake-up of BMA400 to normal mode and/or the wake-up of the external MCU by mapping the interrupt to the interrupt pin.

The wake-up interrupt function evaluates acceleration data and is set as soon as the value of the sampled data exceeds the preconfigured acceleration threshold. The comparison of the current acceleration value with a reference is configurable between relative reference (last sampled value stored in the register) and absolute reference (the reference values are set once and not changing after each acceleration conversion). The delay between two data conversions is 40ms (25Hz conversion ODR in Low-power).

The auto wake-up by wakeup interrupt on activity is activated by setting AUTOWAKEUP_1.wkup_int bit. The wakeup status is available in INT_STAT0.wkup_int.

When woken up, an interrupt can be generated and mapped to the interrupt pins.

The wake-up interrupt function supports following configurations:

- Selectable axis for wake-up: the wake-up interrupt function supports independent activation/deactivation of each acceleration axis for function evaluation. This is performed by setting the bits WKUP_INT_CONGIF0.wkup_X/Y/Z_en accordingly.

- Reference update mode (configured by setting WKUP_INT_CONFIG0.wkup_refu)

wkup_refu<1:0>	description of wake-up interrupt references update mode
b00	manual update The references (int_wkup_refX/Y/Z) are not updated automatically, they shall be set manually by user in low-power mode.
b01	one time The references is updated every time at entering low power mode. The first measured acceleration in Low-power mode is used as reference.
b10 or b11	every time The reference is updated every time after the acceleration conversion in low-power mode

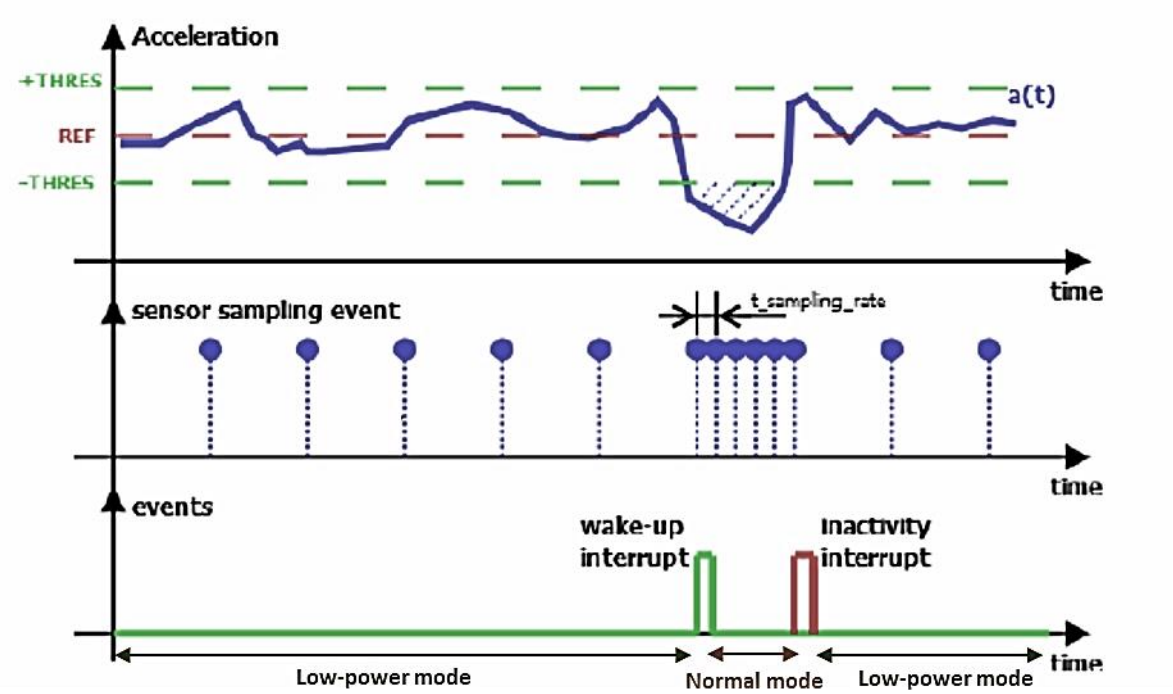
  The reference values are 8-bit signed values. The activity measurement takes the upper 8 bits of the acceleration value and compares against the reference WKUP_INT_CONFIG[2-4].int_wkup_ref[x,y,z].

- Threshold for activity detection: the threshold for activity detection (comparison of the difference between the measured acceleration data and reference acceleration data) has 8-bit resolution, corresponding to the upper 8 bits of the absolute value of the 12bit acceleration, WKUP_INT_CONFIG1.int_wkup_thres.

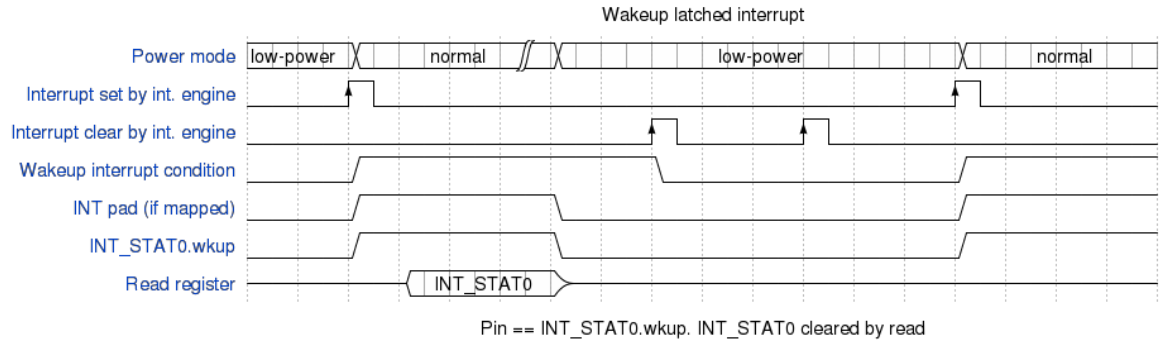
- Number of samples for decision: the number of samples for wake-up decision is configured between 1 and 8 by the register WKUP_INT_CONFIG0. num_of_samples (number of samples is the register value + 1).

The condition for activity-driven automatic wake-up from low-power is (assuming all 3 axes are enabled):
(abs(a_x-ref_x) > thresh_x) OR (abs(a_y-ref_y) > thresh_y) OR(abs(a_z-ref_z) > thresh_z)
This condition must persist for WKUP_INT_CONFIG0. num_of_samples data samples.

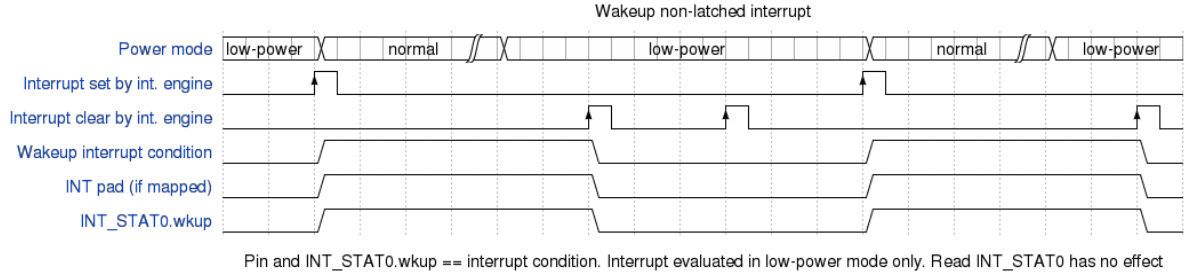
The wake-up on activty is illustrated in the following picture



Wake-up interrupts can be used latched and non-latched (see chapter 0). Latched and non-latched behavior is shown below.



Pin == INT_STAT0.wkup. INT_STAT0 cleared by read



Pin and INT_STAT0.wkup == interrupt condition. Interrupt evaluated in low-power mode only. Read INT_STAT0 has no effect

Auto low-power mode

Power mode can be changed from Normal to Low-power mode through a serial interface command. It is also possible to change automatically (without a serial command) from normal mode to low-power mode, which is called auto low-power.

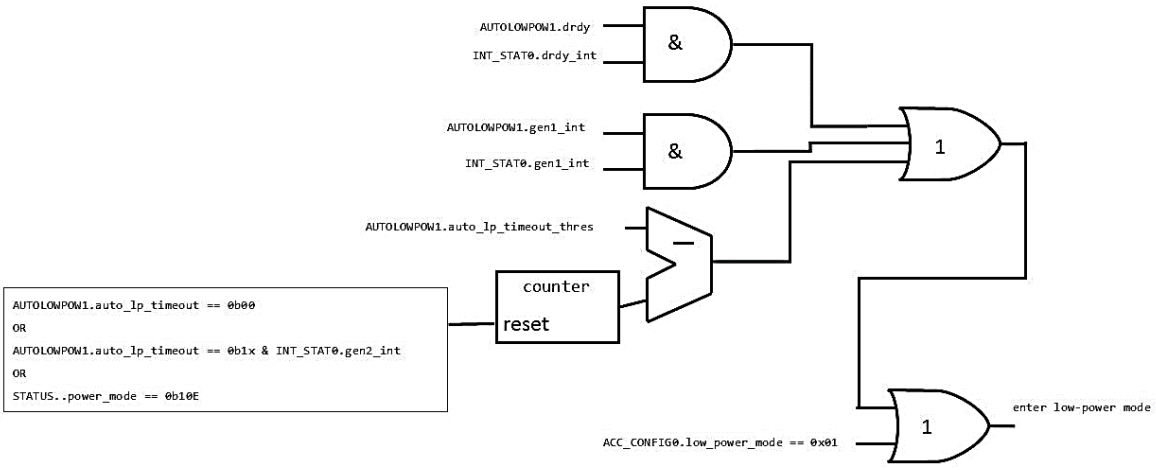
The following timed and non-timed triggers are supported for automatic switching from Normal mode to Low-power mode:

- First data ready: (AUTOLOWPOW_1.auto_lp_timeout =b00)
  If AUTOLOWPOW_1.drdy = '1', BMA400 is set into low-power mode when new data calculation is finished.
- Generic interrupt 1: (AUTOLOWPOW_1.auto_lp_timeout =b00)
  If AUTOLOWPOW_1.gen1_int = '1', BMA400 is set into Low-power mode as soon as the Generic interrupt 1 is detected. (see chapter 4.7)
- low_power_timeout (AUTOLOWPOW_1.auto_lp_timeout =b01): the sensor is set into low-power mode as soon the timeout counter reaches AUTOLOWPOW_1.auto_lp_timeout_thres. The auto-low-power timeout counter is 12 bits wide and is incremented every 2.5ms.
- low_power_timeout with counter reset on activity detected (AUTOLOWPOW_1.auto_lp_timeout =b10,b11): the timeout counter is restarted in case generic interrupt 2 (see chapter 4.7) is asserted.
  The sensor is set into low-power mode when finally the timeout counter reaches AUTOLOWPOW_1.auto_lp_timeout_thres. The auto-low-power timeout counter is 12 bits wide and is incremented every 2.5ms.

The timed timeout trigger can be configured by setting AUTOLOWPOW_1.auto_lp_timeout bits in register according to the table below.

AUTOLOWPOW_1.auto_lp_timeout<1:0>	Description
b00	timeout disabled, use either *AUTOLOWPOW_1.drdy* or *AUTOLOWPOW_1.gen1_int* to switch automatically into low-power mode
b01	timeout active, BMA400 switching into low-power mode as soon as timeout counter reaches AUTOLOWPOW_1.auto_lp_timeout_thres
b10 or b11	Low-power timeout active, timeout counter resets on activity detection

Multiple selections of auto-low-power conditions are supported. Any selected condition switches the device into low-power mode (OR condition). The logical connection of the auto-low-power conditions is shown in the picture below.


AUTOLOWPOW1.drdy
INT_STAT0.drdy_int
&
AUTOLOWPOW1.gen1_int
INT_STAT0.gen1_int
&
1
AUTOLOWPOW1.auto_lp_timeout_thres
counter
reset
AUTOLOWPOW1.auto_lp_timeout == 0b00
OR
AUTOLOWPOW1.auto_lp_timeout == 0b1x & INT_STAT0.gen2_int
OR
STATUS..power_mode == 0b10E
1
ACC_CONFIG0.low_power_mode == 0x01
enter low-power mode

4.4. Sensor Data

Acceleration Data

The width of acceleration data is 12 bits given in two´s complement representation in the registers 0x04 to 0x09 (ACC_X_LSB, ACC_X_MSB, ACC_Y_LSB, ACC_Y_MSB, ACC_Z_LSB, ACC_Z_MSB). The 12 bits for each axis are split into an MSB upper part (4 bit) and an LSB lower part (8 bit).
In order to ensure the integrity of the acceleration data read, the content of all data registers must be read in a single burst read, since these registers are write-protected during a read access. As soon as the burst read is finished the register content will be updated if new data are available.

Filter Configuration

Two major filter paths are implemented, see block diagram. Filter output can either be fed into the data registers, into the FIFO, or used to process interrupts in the interrupt engine. This is selectable by customer.

Filter1 (acc_filt1) has a data rate between 800Hz and 12.5Hz, controlled by ACC_CONFIG0.acc_odr. Its bandwidth can be configured additionally by ACC_CONFIG0.filt1_bw:

- ACC_CONFIG0.filt1_bw = 0x0 →0.48 x ODR
- ACC_CONFIG0.filt1_bw = 0x1 →0.24 x ODR

ACC_CONFIG0.acc_odr <3:0>	Output Data Rate [Hz]
0xB .. 0xF	800
0xA	400
0x9	200
0x8	100
0x7	50
0x6	25
0x0 .. 0x5	12.5

Filter2 (acc_filt2) has a fixed data rate of 100 Hz.
In addition, these 100 Hz data of filter2 can be used by a low pass filter (*acc_filt_lp*) with a bandwidth of 1 Hz. The output data rate will stay at 100 Hz. This data can be used as input for the data registers and also in the interrupt engine. Access via FIFO is not possible.

ACC_CONFIG2.data_src_reg<1:0>	Filter output going into data registers (not FIFO!)
0x0,0x3	acc_filt1(selectable ODR)
0x01	acc_filt2 (48Hz BW, 100Hz ODR)
0x02	acc_filt_lp (1 Hz BW, 100 Hz ODR)

FIFO_CONFIG0.fifo_data_src	Filter output going into FIFO
0x0	acc_filt1(selectable ODR)
0x1	acc_filt2 (100Hz ODR)

In low-power mode, only data at 25Hz ODR is available. Depending on the setting of ACC_CONFIG0.osr_lp, noise and current consumption is controllable.

G-range selection

The measurement g-range can be selected between 2g and 16g. It can be configured ACC_CONFIG1.acc_range.

ACC_CONFIG1.acc_range<1:0>	Selected g-range
11	16g
10	8g
01	4g
00	2g

Data Ready Interrupt

This interrupt fires whenever a new data sample set is complete. This allows a low latency data readout. In non-latched mode, the interrupt and the flag in Register INT_STAT0 are cleared automatically after 1/(1600Hz). If this automatic clearance is unwanted, latched-mode can be used. In order to enable/use the data ready interrupt map it on the desired interrupt pin via INT1_MAP or INT2_MAP.

Temperature Sensor

The temperature sensor has 8 bits resolution. The temperature value is defined in Register TEMP_DATA and updated every 160ms.

It is always on when the sensor is active (in normal and in low-power mode, not in sleep mode).

Value	Temperature
0x7F	87.5 °C
...	...
0x02	25 °C
...	...
0x80	-40.0 °C

The temperature sensor is calibrated with a precision of +/-5°C.

Sensor Time

The BMA400 has an integrated sensor timer. The sensor time can be used for synchronization purposes between the external MCU and the sensor.
The sensor timer counts the clock cycles generated by the system clock which is always running in low-power and normal modes. Sensor timer is inactive in sleep mode and reset when entering the sleep mode. Counter values are stored in registers *SENSOR_TIME(0/1/2)*.
The sensor timer has a resolution of 21 bits stored in 3 bytes. For compatibility with other sensors that use faster counters with 25.6 kHz, the lower three bits of the counter (*sensor_time<2:0>)* are always 0. Thus, the lowest significant bit of the counter is *sensor_time<3>*.
After the timer has reached the maximum value, the counter resets to zero.

Bit *m* in sensor_time	**23**	**22**	**21**	**...**	**8**	**7**	**6**	**5**	**4**	**3**
Resolution	327.68 s	163.84 s	81.92 s	...	10 ms	5 ms	2.5 ms	1.250 ms	0.625 ms	0.3125 ms
Update rate [Hz]	0.0031	0.0061	0.012	...	100	200	400	800	1600	3200

The sensortime is synchronized with the data capturing in the data register and the FIFO. The sensortime supports multiple seconds of sample counting and a sub-millisecond resolution.

Burst reads on the registers SENSORTIME_0 to SENSORTIME_2 deliver always consistent values, i.e. the value of the register does not change during the burst read.

4.5. FIFO

FIFO description

Acceleration data are stored in a 1024Bytes FIFO. The FIFO is written only in normal mode.
When *FIFO_CONFIG0.fifo_stop_on_full* = '0', the device is in stream mode.
When *FIFO_CONFIG0.fifo_stop_on_full* = '1', the device is in FIFO mode.

- Stream mode: overwrites oldest data on FIFO full condition
- FIFO full mode: discards newest data on FIFO full condition

The FIFO depth is 1024 byte and supports the following interrupts:

- FIFO full interrupt
- FIFO watermark interrupt

The data to be collected is defined through *fifo_data_src, fifo_x_en, fifo_y_en* and *fifo_z_en* bits. FIFO is disabled when no writing is defined; FIFO is therefore disabled when *fifo_x_en='0', fifo_y_en='0'* and *fifo_z_en='0'*.

If the FIFO is disabled when FIFO byte count is greater than 0, no new frame is written to the FIFO, but FIFO is operational:

- Frames already written in the FIFO remain stored and can be read out
- FIFO interrupts and their corresponding statuses are still evaluated
- after all bytes are read out, sensortime (if enabled) and empty frames are generated
- FIFO can be flushed

FIFO input data

Storing of acceleration measurement results is enabled by setting respectively *fifo_x_en* = '1' and/or *fifo_y_en = '1'* and/or *fifo_z_en = '1'*. Storing of data can be enabled or disabled on a per-axis basis in any combination.
acc_filt1 or *acc_flit2* data are stored in the FIFO depending on *fifo_data_src bit*.
Thus, the data rate with which data is stored in the FIFO equals the data rate with which the filter serving as data source is configured.

The number of bytes available in the FIFO is readable through *fifo_bytes_cnt<10:0>*.
The FIFO byte count registers *FIFO_LENGTH0* and *FIFO_LENGTH1* are updated only when a full frame has been written to the FIFO and is available for read-out. FIFO byte count registers are also updated after each full frame read from the FIFO.
FIFO byte count registers increment or decrement is equal to the frame length; intermediate increments (corresponding to a partial frame) are not readable.
The FIFO shall support two modes for acceleration data storage in FIFO: 12 bits stored as two bytes into FIFO and 8-bit mode (upper 8 bits of 12 bits) stored as single byte into FIFO per acceleration axis.
The 8-bit mode activation shall be performed by setting *FIFO_CONFIG0.fifo_8bit_en* = '1'.

FIFO read out

The FIFO can be read out via *FIFO_DATA* register in a single burst read, this allows a complete reading of the FIFO content within one burst read transaction.
FIFO read out is not supported in Sleep mode.
FIFO read out is supported in normal and Low-power mode if *FIFO_PWR_CONFIG.fifo_read_dis* = '0'.
The minimum delay T_{fifo_read} = 50µs has to be applied between enabling FIFO read and the start of FIFO read. Don't read the FIFO when *FIFO_PWR_CONFIG.fifo_read_dis* ='1'.

FIFO overflow behavior

A FIFO overflow occurs if the FIFO is full and a new data is to be written to the FIFO. FIFO full means free space is less than maximum frame length of 9 bytes. The largest frame is 7 bytes long, however each time FIFO is written (at the end of the measurement), 9 bytes can be written to the FIFO in total, consisting of 2 frames: one with the measurement results (maximum of 7 bytes), and configuration change frame consisting of 2 bytes. The definition of the full interrupt uses 9 bytes limit to give the host system time to react to it before the FIFO overflows.

In case of overflow the FIFO can either stop recording data or overwrite the oldest data. The behavior is controlled by register *fifo_stop_on_full*.
Streaming mode, *fifo_stop_on_full* = '0': if the new frame does not fit inside the remaining free space in the FIFO RAM, FIFO will repeatedly delete the oldest frame until it creates enough space for the new one.
FIFO stop-on-full mode, *fifo_stop_on_full* = '1': The newest frame is discarded.
Normal operation resumes if the FIFO full condition no longer persists.

Frames

The FIFO captures data in frames, which consist of a header and a payload.

- Each data frame consists of a one byte header describing properties of the frame, (which data are included in this frame) and the data itself. Beside the data frames, there are control frames, sensortime frames and empty frames.

The header has a length of 8 bit and the following format:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	fh_mode<1:0>		fh_param<4:0>					0

fh_mode and fh_param<4> indicate whether the frame is a data frame (accel data), a sensortime frame (sensortime data), a control frame or an empty frame (all data 0).

A data frame is composed of the said header and a set of acceleration data organized as described in table below.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	fh_mode<1:0>		fh_param<4:0>					0
Data 1..7	1 .. 7 Data bytes, number depending of 12 or 8bit storage mode and number of axes enabled.							

These *fh_mode* and *fh_parm* fields are defined below

fh_mode<1:0>	**Definition**	**fh_param <4>**	**fh_param <3>**	**fh_param <2:0>**
0b10	Sensor data frame	b0: Sensor data frame	b0: 8bit mode b1: 12bit mode	Enabled axes
0b10	sensortime frame	b1: sensortime frame	no meaning	No meaning
0b01	Control frame	b00100		

Name	**fh_parm<2:0>**		
Bit	2	1	0
Content	z-enabled	y-enabled	x-enabled

f_param<3:0>=0b0000 is invalid for regular mode, a header of 0x80 indicates an uninitialized frame.
In a data frame, fh_param<2:0> defines which sensors axes are included in the data part of the frame.
fh_param<3> defines in which resolution – 8 or 12bit – the data are stored.
fh_param<2/1/0> indicate whether Z, y or x axis data are stored.
Thus, fh_param<3:0> allows to calculate the amount of data payload following the header.
The maximal payload is 6 bytes if all axes are enabled and 12bits are stored.
3bytes payload are needed if all axes are enabled and 8bits are stored.
A lesser amount of data is required if one or two axes are disabled.

As an example, data frames with 12bit and 8bit resolution are shown below, all axes enabled

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	1	0	0	1: 12bit	1: Z	1: Y	1: X	0
data	unused				acc_x<3:0>			
	acc_x<11:4>							
	unused				acc_y<3:0>			
	acc_y<11:4>							
	unused				acc_z<3:0>			
	acc_z<11:4>							

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	1	0	0	0: 8bit	1: Z	1: Y	1: X	0
data	acc_x<11:4>							
	acc_y<11:4>							
	acc_z<11:4>							

A FIFO empty frame is a sensor data frame, this is what the header indicates (fh_mode=b10).
fh_param<2:0>=b000 shows that the frame delivered is an empty frame and contains 1 data byte of value 0x00 after the header.
This kind of frame is delivered if the last frame in the FIFO was already read out or if the FIFO is empty. The format is shown below.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	1	0	0	0	0	0	0	0
Data	0	0	0	0	0	0	0	0

If fh_param<4:0>= b10000, the header indicates a sensor-time frame to come, its format shown below.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	1	0	1	0	0	0	0	0
time	sensor_time<7:0>							
	sensor_time<15:8>							
	sensor_time<23:16>							

The data for the sensor-time frame consists of registers *sensor_time2/1/0* at the moment the sensor-time frame transmission has started. A sensor-time frame is not stored in the FIFO, it is created on-the-fly and delivered with a FIFO burst read operation when all acceleration data frames have been transmitted and the burst read carries on requesting data.
The sensortime frame will only be delivered if *fifo_time_en* = '1'.

The already mentioned control frame looks as follows

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header	0	1	0	0	1	0	0	0
Opcode	0	1	1	0	0	acc_config1_chg	acc_config0_chg	fifo_config0_chg

- fifo_config0_chg = b1: The control frame will be inserted when *FIFO_CONFIG0.fifo_data_src* change becomes active in FIFO.
- acc_config0_chg = b1: The control frame will be inserted when *ACC_CONFIG0.filt1_bw* change is valid for data stored in FIFO.
- acc_config1_chg: The control frame will be inserted when *ACC_CONFIG1.acc_odr* or *ACC_CONFIG1.osr* or *ACC_CONFIG1.acc_range* change is valid for data stored in FIFO.

If more changes become active at one acceleration sample just one control frame will be inserted, with more than one of the three CONF_chg bits set.

The data format for data frames is identical to the format defined for the data registers: signed integer. If no axis is selected for FIFO storage no frames are written into the FIFO.

Under-read

In case the FIFO is under-read (not all frames were taken from the FIFO, but the last frame read was read entirely), the next readout will continue at the frame that was just about to be sent.

Partial frame read

In case the FIFO is under-read and a partial data frame read occurred (not all frames were taken from the FIFO, and the last frame read was not read entirely), the entire last data frame is repeated upon the next read access.
When *fifo_stop_on_full*='0' oldest frames are overwritten when new frames are available and the FIFO is full.
When this happens, the partially read data frame is not repeated but the oldest frame available in the memory is sent instead.
Sensortime frame is not repeated when it is read only partially.

If the read of a frame is interrupted during the frame's last byte read, this partial read is not recognized and the frame is discarded like a fully read frame.

Over-read

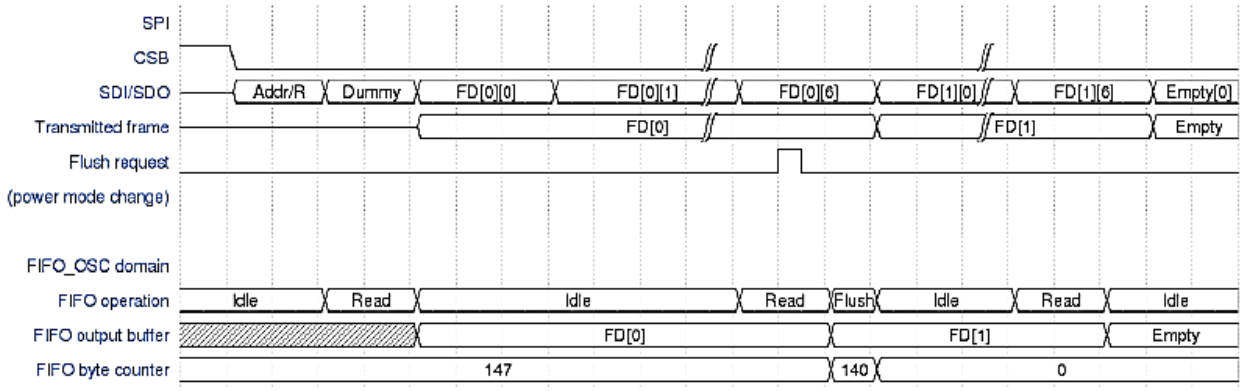
If the burst read continues after all frames have been read out, a sensortime frame is sent after the FIFO becomes empty during a burst read operation if *fifo_time_en*='1'. After that or when FIFO was completely read, the empty frame is returned as long as the burst read is active.

Reading nearly-empty FIFO

FIFO contains a reading cache buffer for a complete frame. When there is only one unread byte left in the reading buffer, the FIFO starts prefetching the next frame from the memory to be ready for burst reading if there is any further frame, or it evaluates itself as empty.
If new data frames/config frames are written to the FIFO before this reading event, the FIFO will behave as containing one further frame and the new frame will be made available for reading as the next frame. If new data/config frames are written to the FIFO after the moment when "only one unread byte is left in the buffer", then user will see the FIFO as empty after the current frame will be finished.

FIFO flushing

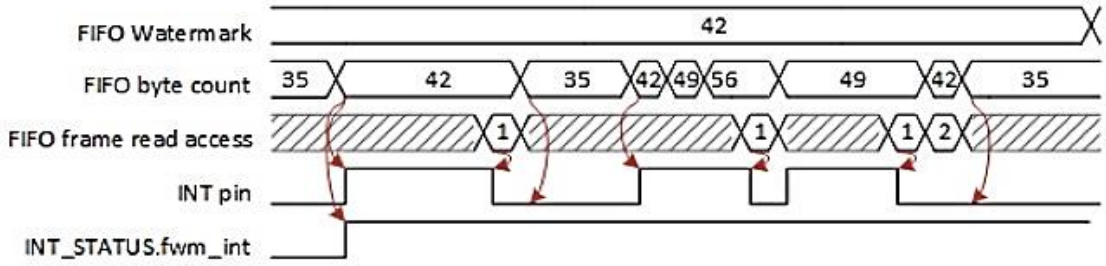
A FIFO flush operation is executed when a *flush* command is written to the CMD register, when a soft-reset command is issued or when the device changes power mode and FIFO auto flush is enabled through *FIFO_CONFIG0.auto_flush* bit. For system simplicity, a flush is executed as soon as possible. FIFO can be written or flushed at any time when FIFO is not read.
Flush operation does not depend on serial interface activity to finish. Power mode transition (or write) does not have to wait for the Flush to finish. Serial interface always reads what is in the FIFO at the moment the next frame is prepared for the output buffer. Empty frames are read if the FIFO was flushed during the transaction.



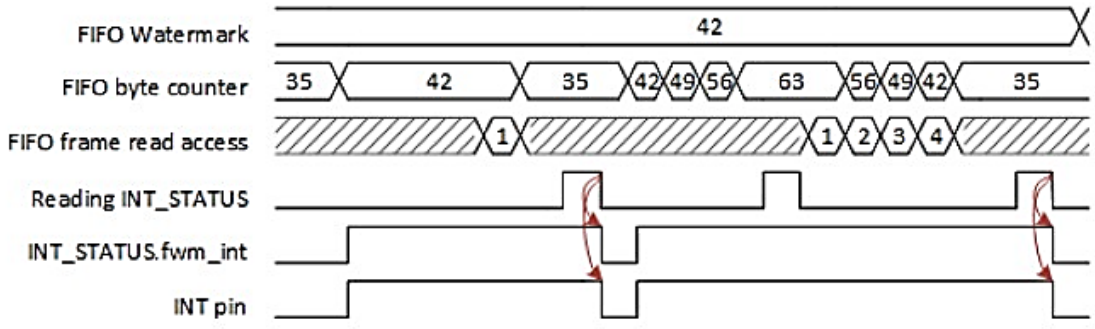
FIFO watermark interrupt

Watermark interrupt status is asserted when the watermark interrupt condition is satisfied i.e. when the filling level of the FIFO (number of unread bytes in the FIFO) is greater or equal to the watermark level (*fifo_bytes_cnt<10:0> ≥ fifo_watermark<10:0>*).
FIFO watermark interrupt is enabled by setting INT_CONFIG0.fwm_int = '1'. It can be mapped to INT1/2 pad.
When the FIFO watermark level is set to zero, the interrupt condition is never satisfied. The status of the watermark interrupt can be read back through the INT_STAT0.fwm_int bit.
Interrupt status is cleared by reading the INT_STAT0.fwm_int bit when the FIFO filling level is lower than the watermark level.

The interrupt is only evaluated after entire frames have been read out or written (as the counter is only in-/decreased on a frame basis).
Watermark interrupt condition is also updated after the end of the serial interface (burst write) transaction which wrote into the registers *fifo_watermark<10:8> or fifo_watermark<7:0>*.
The behavior of the FIFO watermark is shown in the figures below.



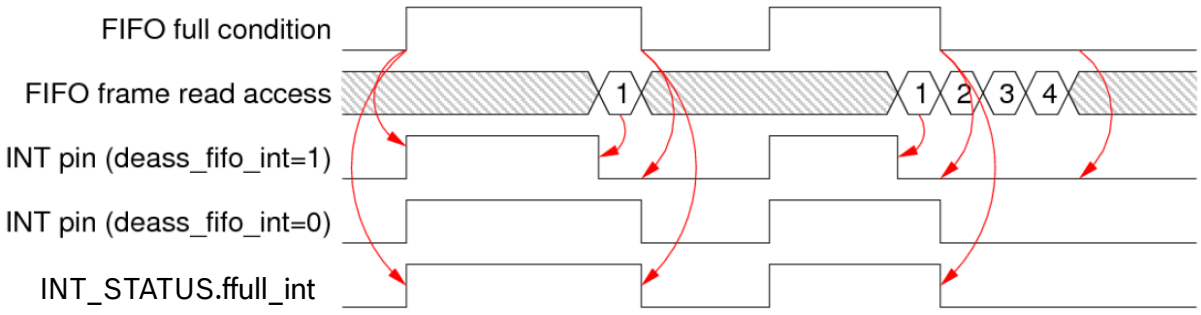
FIFO watermark interrupt, non-latched, with reads from FIFO



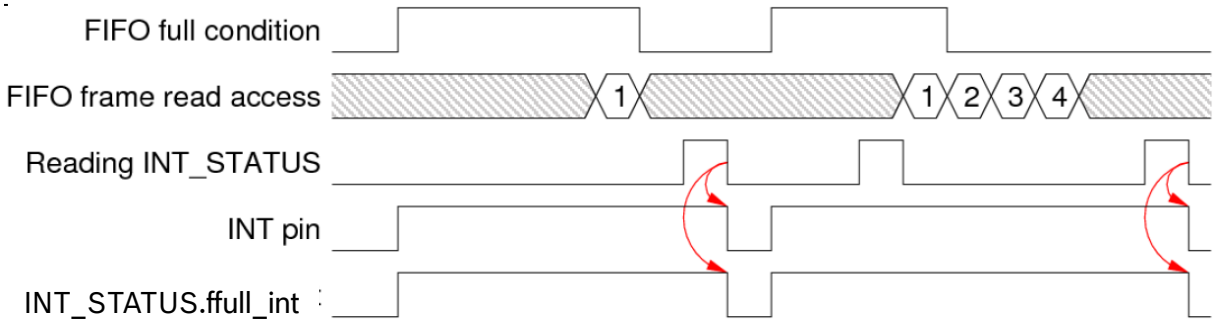
FIFO watermark interrupt, latched, with reads from FIFO

FIFO full interrupt

The full interrupt can be enabled by setting bit INT_CONFIG0.*ffull_en* = '1'. It can be mapped to INT1/2 pad. Full interrupt status is asserted when the full interrupt condition is satisfied, when the filling level of the FIFO (number of unread bytes in the FIFO = *fifo_bytes_cnt<10:0>*) is equal or higher than 1016. The status of the full interrupt can be read back through the INT_STAT0.*ffull_int* bit.
Interrupt status is cleared by reading the INT_STAT0.*ffull_int* bit high '1' when the FIFO filling level is lower than 1016. The behavior of the FIFO full interrupt is shown in the figures below.



FIFO full interrupt, non-latched, with reads from FIFO



FIFO full interrupt, latched, with reads from FIFO

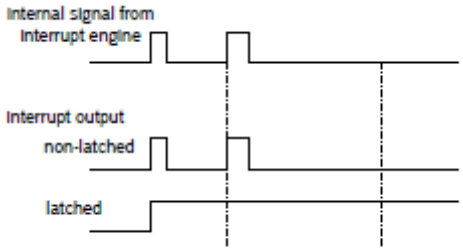
4.6. General Interrupt Pin configuration

Interrupt Pin Mapping

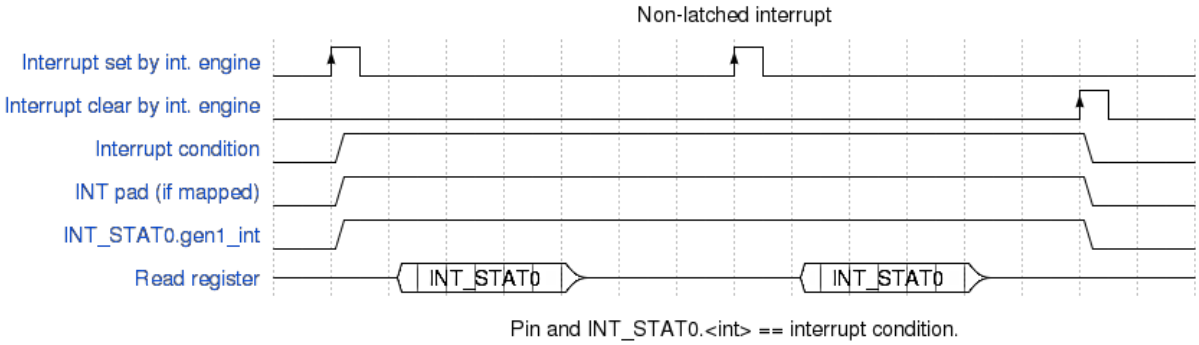
The content of the interrupt status registers can be mapped to pins INT1 or INT2, by setting the corresponding bits from the registers INT1_MAP, respectively INT2_MAP or INT12_MAP.
To disconnect the features outputs to the external pins, the same corresponding bits must be reset, from the registers, INT1_MAP, respectively INT2_MAP or INT12_MAP.
Once a feature triggered the output pin, the Host can read out the corresponding bit from the register, INT_STAT0, INT_STAT1 or INT_STAT2 .

Interrupt latching

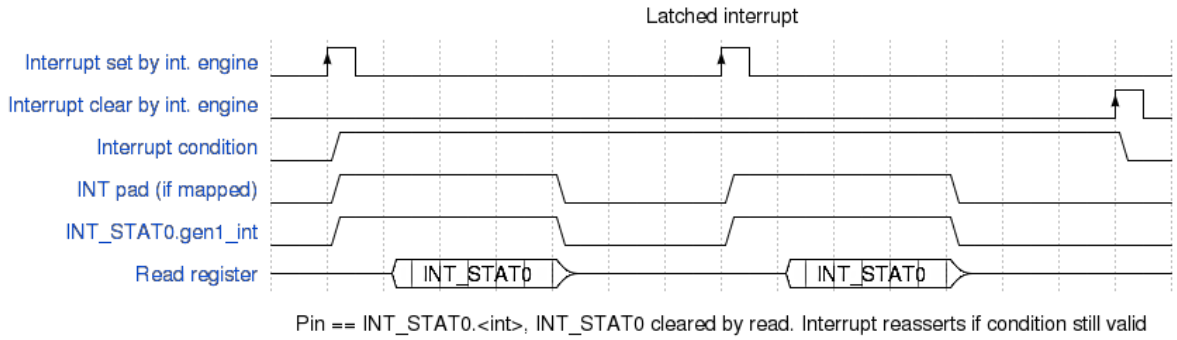
Interrupts can be configured as non-latched or latched. The mode is selected by INT_CONFIG1.latch_int. Latching determines when an interrupt is released. The behavior of the different interrupt modes is shown graphically in the figure below



Non-latched mode
In the non-latched mode (INT_CONFIG1.latch_int = 0), both the INT pins (the contribution to the ´or´ condition for the INT pin) and the interrupt status bit in INT_STAT are reset when the interrupt activation condition is released.



Latched mode
In latched mode (INT_CONFIG1.latch_int = 1) an asserted interrupt status in INT_STAT(0/1/2) and the INT pin (the contribution to the ´or´ condition for the INT pin) is cleared by reading the corresponding status register. If the FIFO filling activation condition still holds true then the interrupt status is not cleared. Data ready and advanced interrupts' statuses are cleared upon reading INT_STAT register.



Interrupt behavior during power mode switching

When the device leaves normal mode, all internal interrupt status registers are cleared. There are two exceptions:

- The step counter keeps its state (i.e. the step count) on mode switching. If the mode is switched to normal with enabled step counter, it continues counting on the previous value. The internal interrupt status is cleared.
- FIFO interrupts are not cleared by mode switching

Electrical Interrupt Pin Behavior

Both interrupt pins INT1 and INT2 can be configured to show the desired electrical behavior.
The 'active' level of each interrupt pin is determined by the *int1_lvl* and *int2_lvl* bits.
If *int1_lvl* = 1 / *int2_lvl* = 1, then pin "INT1" / pin "INT2" are active HIGH.
The characteristic of the output driver of the interrupt pins is configured with bits *int1_od* and *int2_od*.
By setting bits *int1_od* / *int2_od* to '1', the output drivers show open-drive characteristic, by setting the configuration bits to 0, the output drivers show CMOS push-pull characteristic.
When open-drive characteristic is selected in the design, an external pull-up or pull-down resistor should be applied according the *int(1/2)_lvl* configuration.
For all interrupts, the user is responsible of the settings, no hardware checks of the settings are implemented before processing interrupts.

int(1/2)_od	int(1/2)_lvl	"INT1" / "INT2"	output driver
0	0	active '0'	push-pull characteristic
0	1	active '1'	push-pull characteristic
1	0	active '0'	open-drive characteristic sink (NMOS)
1	1	active '1'	open-drive characteristic source (PMOS)

4.7. Interrupt Features

The following interrupts exist in the BMA400:

Basic interrupts
- Data ready interrupt
- FIFO watermark
- FIFO full
- Interrupt engine overrun
- Wake-up interrupt

Advanced Interrupts
- Generic interrupt 1
- Generic interrupt 2
- Step detector interrupt/step counter
- Activity changed interrupt
- Single tap / Double tap sensing
- Orientation changed interrupt

Basic interrupts can all be enabled independently from each other.

Advanced interrupts are only available in normal mode, the interrupt engine is disabled in low power mode and sleep mode. Advanced interrupts are served by the interrupt engine. They share the same resources, thus, enabling too many interrupts of this type in parallel may lead to a so-called Interrupt engine overrun. This interrupt indicating that the interrupt engine could not finish calculating all selected interrupt conditions.
If this occurs, advanced interrupts of lesser importance must be disabled until the Interrupt engine overrun condition/interrupt vanishes.

Any change of an interrupt configuration must be executed when the corresponding interrupt is disabled.

For most interrupts a data rate of 100Hz is recommended, only tap sensing requires 200Hz. It is then necessary to configure the data source of the tap sensing interrupt, filter acc_filt1, to 200Hz, which implies that the other interrupts requiring 100Hz data rate use another filter.

Interrupt pin mapping, interrupt status

The BMA400 supports flexible INT1 and INT2 pin mapping configurations via interrupt mapping registers INT1_MAP, INT2_MAP and INT12_MAP. Depending on these registers settings, all interrupt sources are mapped to the INT1 and INT2 pins.
The status of the interrupts can be read out at the status registers INT_STAT0, INT_STAT1 and INT_STAT2.
Additionally, the step counter value is stored in the registers STEP_CNT0...STEP_CNT2 . These registers need to be read out using a burst read to avoid one register being updated while another step count register is read.

Generic Interrupt 1 and 2

The generic interrupts 1 and 2 have the exact same implementation. They are designed to detect activity or inactivity.

The generic interrupt monitors acceleration change with respect to a reference, or in other words, the difference between actual acceleration and reference is calculated and compared against a threshold. The comparison is de-noised using a hysteresis.

The generic interrupt is triggered when the above mentioned difference lasts for a minimum time.

Reference, threshold, hysteresis and duration are configurable.

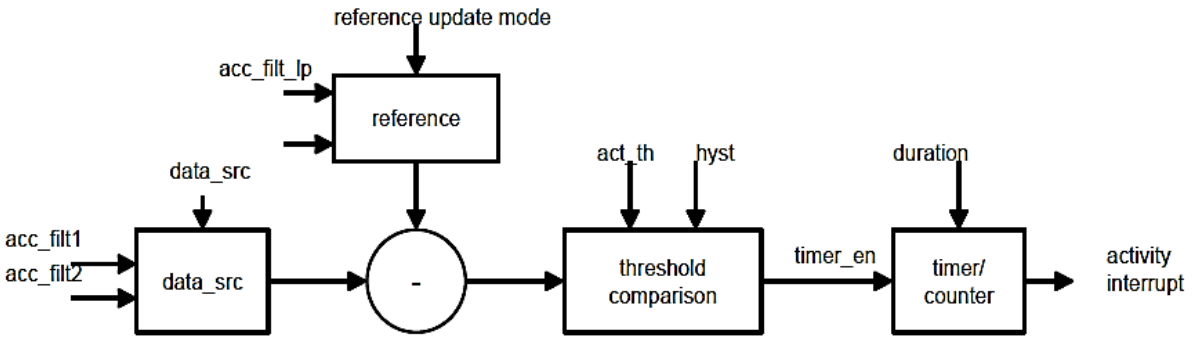
Both generic interrupts work the same way, but have separate sets of registers to be processed independently of each other.

- Generic interrupt 1 is enabled by 'INT_CONFIG0.*gen1_int_en = 1'*
- Generic interrupt 2 is enabled by 'INT_CONFIG0.*gen2_int_en = 1'*

The generic interrupt supports selectable acceleration axes for evaluation:
GEN(1/2)INT_CONFIG0.*act_(x/y/z)_en*.

GEN(1/2)INT_CONFIG1.comb_sel selects if the interrupt shall be based on an AND (*comb_sel = 1)* or an OR (*comb_sel = 0)* combination of all enabled axes.

The acceleration data source is selectable between acceleration from *acc_filt1* or *acc_filt2* by setting GEN1/2INT_CONFIG0.*data_src* (0: acc_filt1, 1:acc_filt2).



The data rate for the filter output must be 100Hz. Using acc_filt2 is recommended. In this case acc_filt1 can be used independently from the interrupt engine for the data output registers and the FIFO.

GEN(1/2)INT_CONFIG0.data_src	Source for generic interrupt data
0	acc_filt1
1	acc_filt2

The mentioned reference can be static (user defined) or it can be updated dynamically.
The reference acceleration registers support reference update modes after comparison evaluation has been done. The mode is set in GEN(1/2)INT_CONFIG0.act_refu

GEN(1/2)INT_CONFIG0.act_refu	Description of reference update mode
b00	manual update – reference is statically set by user using GEN(1/2)INT_CONFIG4/5/6/7/8/9 (read back after interrupt is activated)
b01	one time – the reference is updated once by acceleration data taken from the data source (*acc_filt1 or acc_filt2*) **after triggering the** interrupt Reference will also be updated after entering normal mode.
b10	every time – the reference is updated at the end of the interrupt evaluation, it is taken from the data source (*acc_filt1 or acc_filt2*). This mode especially makes sense for activity detection where a “constantly” increasing acceleration shall be detected.
b11	every time - the reference is updated at the end of the interrupt evaluation, it is taken from the data source *acc_filt_lp*. Remember the large group delay (1Hz bandwidth) **of** acc_filter_lp

As already mentioned, both interrupts can be configured to detect activity or inactivity. This is done using GEN(1/2)INT_CONFIG1.criterion_sel.

GEN(1/2)INT_CONFIG1.criterion_sel = 0: inactivity detection, referenced acceleration below threshold
GEN(1/2)INT_CONFIG1.criterion_sel = 1: activity detection, referenced acceleration above threshold

The reference values for each axis are stored in registers GEN(1/2)INT_CONFIGX.int_th_ref(x/y/z), they are 12-bit signed values. The values are range sensitive, and the physical value of LSB corresponds to 2^(2+acc_range)/4096, here acc_range = 0, 1, 2, 3 corresponds to +/-2g, +/-4g, +/-8g, +/-16g.
The threshold value are stored in register *GEN(1/2)INT_CONFIGX.gen_int_thres*, it is 8-bit unsigned value, fixed resolution of 8mg for all measurement ranges.

The interrupt supports a configurable duration condition: *GEN(1/2)INT_CONFIGX.gen1_int_dur<15:0>* indicates the resolution in data ready ticks. So, the duration depends on the data rate the selected filter delivers.
A hysteresis helps to suppress noise in the decision-making. The effective threshold value is set to (*gen_int_thres – act_hyst*) after the first true evaluation of the threshold condition and changed back to (*gen_int_thres*) as soon as the condition is false. *GEN(1/2)INT_CONFIG0.act_hyst*. Following hysteresis configurations for the activity comparison are available:

GEN(1/2)INT_CONFIG0.act_hyst	Description of hysteresis amplitude (mg)
b00	0
b01	24
b10	48
b11	96

Step Detector / Step Counter

The Step Counter algorithm is optimized to high accuracy, while Step Detector is optimized to low latency. Both are running in parallel, once enabled.
The step counter computation is enabled if *INT_CONFIG1.step_int* = '1'.

Step Counter:
The step counter accumulates the steps detected by the step detector interrupt, and makes available the 24 bit current step counter value in the 3 registers STEP_CNT0... STEP_CNT2 , each holding 8bit.

Step Detector:
Once a step is detected the *INT_STAT1.step_int* interrupt signal is set to 1 (one step detected) or 2 (step detected and very likely to have missed a step before; "double step").
There are situations when the step counting value is different than the sum of steps detected by the step detector.

Step Counter/Detector sensitivity:
The step counter uses 24 configuration registers *STEP_COUNTER_CONFIG0 to STEP_COUNTER_CONFIG23.* The step counter and detector use case (device position) can be modified by setting these parameters to the corresponding values in the table below. By default and after reset, the wrist use case is configured.

Register name	Address	wrist (default)	non-wrist
STEP_COUNTER_CONFIG0	0x59	1	1
STEP_COUNTER_CONFIG1	0x5A	45	50
STEP_COUNTER_CONFIG2	0x5B	123	120
STEP_COUNTER_CONFIG3	0x5C	212	230
STEP_COUNTER_CONFIG4	0x5D	68	135
STEP_COUNTER_CONFIG5	0x5E	1	0
STEP_COUNTER_CONFIG6	0x5F	59	132
STEP_COUNTER_CONFIG7	0x60	122	108
STEP_COUNTER_CONFIG8	0x61	219	156
STEP_COUNTER_CONFIG9	0x62	123	117
STEP_COUNTER_CONFIG10	0x63	63	100
STEP_COUNTER_CONFIG11	0x64	108	126
STEP_COUNTER_CONFIG12	0x65	205	170
STEP_COUNTER_CONFIG13	0x66	39	12
STEP_COUNTER_CONFIG14	0x67	25	12
STEP_COUNTER_CONFIG15	0x68	150	74
STEP_COUNTER_CONFIG16	0x69	160	160
STEP_COUNTER_CONFIG17	0x6A	195	0
STEP_COUNTER_CONFIG18	0x6B	14	0
STEP_COUNTER_CONFIG19	0x6C	12	12
STEP_COUNTER_CONFIG20	0x6D	60	60
STEP_COUNTER_CONFIG21	0x6E	240	240
STEP_COUNTER_CONFIG22	0x6F	0	1
STEP_COUNTER_CONFIG23	0x70	247	0

The parameters for wrist use case and non-wrist use case have been obtained using hundreds of experiments used to tweak these parameters for optimal performance. Changing these parameters

should only be done by experts. The reset parameters can be overwritten before enabling the step counter/interrupt.

The step count value is reset during power-on-reset, soft-reset, or step counter reset command transmitted to the device via the command Register CMD.
The step count value is not reset when the step counter is enabled or disabled.

Activity recognition

The step counter state is indicated in the register STEP_*STAT.step_stat_field<1:0>, it* contains the status STILL(0x00), WALK(0x01) or RUN(0x02).

Activity changed interrupt

The device provides an “activity changed” interrupt. The activity changed interrupt evaluates acceleration data for a certain activity over a predefined observation period and sets an interrupt after activity change is detected compared to previously evaluated activity.
The enable signal for this interrupt is *INT_CONFIG1.actch_int*.
The activity changed interrupt supports data source selection by setting *ACTCH_CONFIG1.actch_data_src* bit. The acceleration data source shall be selectable between acceleration from *acc_filt1* and acceleration *acc_filt2*.

data_src	Description
0	acc_filt1
1	acc_filt2

Following steps are performed for activity changed interrupt evaluation:

- Evaluation of the current activity parameter: average (| acc - acc_lp |) of the dynamic acceleration with respect to the quasi-static acceleration (low-pass filtered value acc_filt_lp) over a certain observation period.
- Comparison of the currently evaluated activity parameter with last stored activity parameters (activity parameters for previous observation period): abs(curr_value – last_value) > threshold.
- Update / store the activity parameters: curr_value => last_value.
- Activity changed status bits (actch_z_int, actch_y_int, actch_x_int): signalize activity changed for corresponding axes, “1” for activity changed.

Following configurations are supported for activity changed interrupt:

- Selectable acceleration axis for evaluation (*actch_x_en, actch_y_en, actch_z_en*)
- Threshold for activity change. The configuration of the activity threshold is defined by *ACTCH_CONFIG0.actch_thres<7:0>*
- Number of samples of the observation duration. The observation period is defined by the number of data samples used for the evaluation of the activity parameters. The observation period is defined by the setting ACTCH_CONFIG1.actch_npts.

ACTH_CONFIG1.actch_npts<3 :0>	Number of samples for observation
0000	32
0001	64
0010	128
0011	256
0100 .. 1111	512

Tap Sensing Interrupt

The tap interrupt is operating on an input data rate of 200Hz.
It can detect single and double taps. For configuration, there are the registers *TAP_CONFIG* and *TAP_CONFIG_1*.

(TAP_CONFIG. tap_sensitivity) allows to modify the threshold for the minimum tap amplitude (TAP_CONFIG_1. quiet) and (TAP_CONFIG_1. quiet_dt) allow to define the duration of quiet times between double taps and between taps.

acc_filt1 is the data source for the tap interrupt, so, this filter must be configured to 200Hz ODR if this interrupt shall be enabled. There are two different interrupts that can be enabled separately: single tap (*INT_CONFIG1.s_tap_int*) and double tap detection (*INT_CONFIG1.d_tap_int*).

The status of the interrupts is available in *INT_STAT1.s_tap_int* and *INT_STAT1.d_tap_int*.

With *INT12_MAP.tap_int1* the logical OR of both interrupt statuses can be mapped to the INT1 pin.
INT12_MAP.tap_int2 does the same for the INT2 pin.

Config Register	Comment
TAP_CONFIG. tap_sensitivity[2:0] reset default: “000”	modifies the threshold for the minimum tap amplitude The three bits form an unsig**n**ed integer (‘d0.. ‘d7)
TAP_CONFIG. sel_axis[1:0] reset default: “00”	Modifies the selection of the data provided to the algorithm If TAP_CONFIG.sel_axis ==”00” use Z axis data If TAP_CONFIG.sel_axis ==”01” use Y axis data If TAP_CONFIG.sel_axis ==”1X” use X axis data
TAP_CONFIG_1. quiet[3:2] reset default: “01”	QUIET_TIME = ‘d60 if TAP_CONFIG_1.quiet==”00” QUIET_TIME = ‘d 80 if TAP_CONFIG_1.quiet==”01” QUIET_TIME = ‘d 100 if TAP_CONFIG_1.quiet==”10” QUIET_TIME = ‘d 120 if TAP_CONFIG_1.quiet==”11”
TAP_CONFIG_1. quiet_dt[5:4] reset default: “00”	QUIET_TIME_DT = ‘d4 if TAP_CONFIG_1.quiet_dt==”00” QUIET_TIME_DT = ‘d 8 if TAP_CONFIG_1.quiet_dt==”01” QUIET_TIME_DT = ‘d 12 if TAP_CONFIG_1.quiet_dt==”10” QUIET_TIME_DT = ‘d 16 if TAP_CONFIG_1.quiet_dt==”11”
TAP_CONFIG_1. tics_th[1:0] reset default: “10”	TICS_TH= ‘d6 if TAP_CONFIG_1.tics_th=="00" TICS_TH= ‘d9 if TAP_CONFIG_1.tics_th=="01" TICS_TH= ‘d12 if TAP_CONFIG_1.tics_th=="10" TICS_TH= ‘d18 if TAP_CONFIG_1.tics_th=="11"

Interrupt engine overrun

The interrupt overrun is asserted if filter and interrupt computations cannot be finished in a sample acquisition time. The interrupt status is mapped (mirrored) to all interrupt registers *INT_STAT0*, *INT_STAT1* and *INT_STAT2*, bit ieng_overrun_stat. The interrupt is cleared by reading of any of these registers.
The interrupt is mapped to pads INT1 and INT2 by the registers *INT1_MAP. ieng_overrun_int1* and *INT2_MAP. ieng_overrun_int2*.
The interrupt behavior is not dependent on non-latch, latch mode setting.
If this occurs, advanced interrupts of less importance must be disabled until the Interrupt engine overrun vanishes.

Orientation change interrupt

The orientation-change interrupt is enabled by (*INT_CONFIG0.orientch_int*) *= 1*.
The interrupt is optimized to detect a (screen) orientation change when the product is used in a wearable device or similar application.
The orientation change is evaluated by monitoring the acceleration change in X/Y/Z direction (each individually selectable). The orientation change is evaluated as difference to the last stable orientation stored in the reference registers.
The orientation changed interrupt is generated as soon as the orientation change condition is fulfilled on one of the enabled axes selected by (*ORIENTCH_CONFIG0.orient_X/Y/Z_en*).

The orientation change interrupt supports two input acceleration data streams for evaluation: acc_filt2; and the low-pass filtered data source with 1Hz cut-off frequency acc_filt_lp

ORIENTCH_CONFIG0.data_src	Data source for Interupt
0	acc_filt2
1	acc_filt_lp

The threshold for the orientation change interrupt can be configured in the register ORIENTCH_CONFIG1.orient_thres. The threshold configuration has 8 bits and a resolution of 8mg/LSB. In case the acceleration is above the reference acceleration stored for last position for defined period of time ORIENTCH_CONFIG3.orient_dur; the BMA400 orientation change condition is true.

The minimum duration of a new orientation (which shall trigger an interrupt) can be configured in the register ORIENTCH_CONFIG3.orient_dur. The duration register has 8 bits and a resolution of 10ms/LSB.

When the duration condition is fulfilled, the reference orientation is updated according to the configuration stored in the Register (ORIENTCH_CONFIG0.orient_refu). The reference update mode supports following modes:

- no automatic update at all, the reference orientation will be updated by the user when needed
- update with output from filter acc_filt2
- update with output from filter acc_filt_lp

Summarized, the orientation changed interrupt supports following configuration:

- Axis selection for orientation evaluation
- Data source for data evaluation
  - acc_filt2
  - acc_filt_lp
- Thresholds
  - Threshold for orientation change: 8 bits, 8 mg/lsb resolution
  - Duration for stable orientation: 8bits, 10ms/lsb resolution
- Reference update mode:
  - manual update, it is set by user. The values are range sensitive, and the physical value of lsb corresponds to 2^(2+acc_range)/4096, here acc_range = 0, 1, 2, 3 corresponds to +/-2g, +/-4g, +/-8g, +/-16g.
  - update with acc_filt2 value, the reference orientation is updated with current acceleration value as soon as stable orientation is detected

- update with acc_filt_lp value, the reference orientation is updated with current acceleration value as soon as stable orientation is detected

4.8. Sensor Self-Test

The BMA400 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 4g.

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) excitations: SELF_TEST.self_test_sign= b0, b1 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 BMA400.

	x-axis signal	y-axis signal	z-axis signal
BMA400	1500 mg	1200 mg	250 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. Disable all interrupts which could be triggered by self-test activity, this is no hard requirement
2. Enable accelerometer with OSR=3, normal mode.
3. Set ±4g range
4. Set ODR=100Hz, use acc_filt1 output
5. Wait for > 2 ms
6. Enable self-test for all axes and set positive self-test excitation (SELF_TEST.self_test_sign= b0, SELF_TEST.self_test_en_x/y/z = b1)
7. Wait for > 50ms
8. Read and store acceleration + positive excitation values of each axis of interest
9. Change to negative excitation by setting negative self-test exciation SELF_TEST.self_test_sign= b1
10. Wait for > 50ms
11. Read and store acceleration + negative excitation value of each axis of interest
12. Calculate difference of measured acceleration values from steps 8 and 11
13. Disable self-test for all axes: SELF_TEST.self_test_en_x/y/z = b0, SELF_TEST.self_test_sign= b0
14. Wait 50ms before re-enabling interrupts

By subtracting values with both contain the acceleration part (gravity), what remains is the pseudo-acceleration value caused by the self-test excitations.

4.9. 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

Addr (hex)	RegName	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Rst val	Access mode
0x00	CHIPID	chipid<7:0>								0x90	R
0x01	reserved	reserved								0x88	R
0x02	ERR_REG							cmd_err		0x00	R
0x03	STATUS	drdy			cmd_rdy		power_mode<1:0>		int_active	0x00	R
0x04	ACC_X_LSB	acc_x<7:0>								0x00	R
0x05	ACC_X_MSB	0	0	0	0	acc_x<11:8>				0x00	R
0x06	ACC_Y_LSB	acc_y<7:0>								0x00	R
0x07	ACC_Y_MSB	0	0	0	0	acc_y<11:8>				0x00	R
0x08	ACC_Z_LSB	acc_z<7:0>								0x00	R
0x09	ACC_Z_MSB	0	0	0	0	acc_z<11:8>				0x00	R
0x0A	SENSOR_TIME0	sensor_time<7:0>								0x00	R
0x0B	SENSOR_TIME1	sensor_time<15:8>								0x00	R
0x0C	SENSOR_TIME2	sensor_time<23:16>								0x00	R
0x0D	EVENT								por_detected	0x00	R
0x0E	INT_STAT0	drdy_int	fwm_int	ffull_int	ieng_overrun	gen2_int	gen1_int	orientch_int	wkup_int	0x00	R
0x0F	INT_STAT1				ieng_overrun	d_tap_int	s_tap_int	step_int<1:0>		0x00	R
0x10	INT_STAT2				ieng_overrun		actch_z_int	actch_y_int	actch_x_int	0x00	R
0x11	TEMP_DATA	temp_data<7:0>								0x00	R
0x12	FIFO_LENGTH0	fifo_bytes_cnt<7:0>								0x00	R
0x13	FIFO_LENGTH1						fifo_bytes_cnt<10:8>			0x00	R
0x14	FIFO_DATA	fifo_data<7:0>								0x00	R
0x15	STEP_CNT0	step_cnt<7:0>								0x00	R
0x16	STEP_CNT1	step_cnt<15:8>								0x00	R
0x17	STEP_CNT2	step_cnt<23:16>								0x00	R
0x18	STEP_STAT							step_stat<1:0>		0x00	R
0x19	ACC_CONFIG0	filt1_bw	osr_lp<1:0>					power_mode<1:0>		0x00	RW
0x1A	ACC_CONFIG1	acc_range<1:0>		osr<1:0>		acc_odr<3:0>				0x49	RW
0x1B	ACC_CONFIG2					data_src_reg				0xE0	RW
0x1F	INT_CONFIG0	drdy_int	fwm_int	ffull_int		gen2_int	gen1_int	orientch_int		0x00	RW
0x20	INT_CONFIG1	latch_int			actch_int	d_tap_int	s_tap_int		step_int	0x00	RW

5.1. **Register map**

Addr (hex)	RegName	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Rst val	Access mode
0x21	INT1_MAP	drdy_int1	fwm_int1	ffull_int1	ieng_overrun_int1	gen2_int1	gen1_int1	orientch_int1	wkup_int1	0x00	RW
0x22	INT2_MAP	drdy_int2	fwm_int2	ffull_int2	ieng_overrun_int2	gen2_int2	gen1_int2	orientch_int2	wkup_int2	0x00	RW
0x23	INT12_MAP	actch_int2	tap_int2		step_int2	actch_int1	tap_int1		step_int1	0x00	RW
0x24	INT12_IO_CTRL		int2_od	int2_lvl			int1_od	int1_lvl		0x00	RW
0x25										0x00	
0x26	FIFO_CONFIG0	fifo_z_en	fifo_y_en	fifo_x_en	fifo_8bit_en	fifo_data_src	fifo_time_en	fifo_stop_on_full	auto_flush	0x00	RW
0x27	FIFO_CONFIG1	fifo_watermark<7:0>								0x00	RW
0x28	FIFO_CONFIG2						fifo_watermark<10:8>			0x00	RW
0x29	FIFO_PWR_CONFIG								fifo_read_dis	0x00	RW
0x2A	AUTOLOWPOW_0	auto_lp_timeout_thres<11:4>								0x00	RW
0x2B	AUTOLOWPOW_1	auto_lp_timeout_thres<3:0>				auto_lp_timeout<1:0>		gen1_int	drdy	0x00	RW
0x2C	AUTOWAKEUP_0	wakeup_timeout_thres<11:4>								0x00	RW
0x2D	AUTOWAKEUP_1	wakeup_timeout_thres<3:0>					wkup_timeout	wkup_int		0x00	RW
0x2F	WKUP_INT_CONFIG0	wkup_z_en	wkup_y_en	wkup_x_en	num_of_samples<2:0>			wkup_refu<1:0>		0x00	RW
0x30	WKUP_INT_CONFIG1	int_wkup_thres<7:0>								0x00	RW
0x31	WKUP_INT_CONFIG2	int_wkup_refx<7:0>								0x00	RW
0x32	WKUP_INT_CONFIG3	int_wkup_refy<7:0>								0x00	RW
0x33	WKUP_INT_CONFIG4	int_wkup_refz<7:0>								0x00	RW
0x35	ORIENTCH_CONFIG0	orient_z_en	orient_y_en	orient_x_en	data_src	orient_refu<1:0>		0x0 *(required)*		0x00	RW
0x36	ORIENTCH_CONFIG1	orient_thres<7:0>								0x00	RW
0x38	ORIENTCH_CONFIG3	orient_dur<7:0>								0x00	RW
0x39	ORIENTCH_CONFIG4	int_orient_refx<7:0>								0x00	RW
0x3A	ORIENTCH_CONFIG5					int_orient_refx<11:8>				0x00	RW
0x3B	ORIENTCH_CONFIG6	int_orient_refy<7:0>								0x00	RW
0x3C	ORIENTCH_CONFIG7					int_orient_refy<11:8>				0x00	RW
0x3D	ORIENTCH_CONFIG8	int_orient_refz<7:0>								0x00	RW
0x3E	ORIENTCH_CONFIG9					int_orient_refz<11:8>				0x00	RW

Addr (hex)	RegName	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Rst val	Access mode
0x3F	GEN1INT_CONFIG0	act_z_en	act_y_en	act_x_en	data_src	act_refu<1:0>		act_hyst<1:0>		0x00	RW
0x40	GEN1INT_CONFIG1							criterion_sel	comb_sel	0x00	RW
0x41	GEN1INT_CONFIG2	gen_int_thres<7:0>								0x00	RW
0x42	GEN1INT_CONFIG3	gen_int_dur<15:8>								0x00	RW
0x43	GEN1INT_CONFIG31	gen_int_dur<7:0>								0x00	RW
0x44	GEN1INT_CONFIG4	int_th_refx<7:0>								0x00	RW
0x45	GEN1INT_CONFIG5					int_th_refx<11:8>				0x00	RW
0x46	GEN1INT_CONFIG6	int_th_refy<7:0>								0x00	RW
0x47	GEN1INT_CONFIG7					int_th_refy<11:8>				0x00	RW
0x48	GEN1INT_CONFIG8	int_th_refz<7:0>								0x00	RW
0x49	GEN1INT_CONFIG9					int_th_refz<11:8>				0x00	RW
0x4A	GEN2INT_CONFIG0	act_z_en	act_y_en	act_x_en	data_src	act_refu<1:0>		act_hyst<1:0>		0x00	RW
0x4B	GEN2INT_CONFIG1							criterion_sel	comb_sel	0x00	RW
0x4C	GEN2INT_CONFIG2	gen_int_thres<7:0>								0x00	RW
0x4D	GEN2INT_CONFIG3	gen_int_dur<15:8>								0x00	RW
0x4E	GEN2INT_CONFIG31	gen_int_dur<7:0>								0x00	RW
0x4F	GEN2INT_CONFIG4	int_th_refx<7:0>								0x00	RW
0x50	GEN2INT_CONFIG5					int_th_refx<11:8>				0x00	RW
0x51	GEN2INT_CONFIG6	int_th_refy<7:0>								0x00	RW
0x52	GEN2INT_CONFIG7					int_th_refy<11:8>				0x00	RW
0x53	GEN2INT_CONFIG8	int_th_refz<7:0>								0x00	RW
0x54	GEN2INT_CONFIG9					int_th_refz<11:8>				0x00	RW
0x55	ACTH_CONFIG0	acth_thres<7:0>								0x00	RW
0x56	ACTH_CONFIG1	actch_z_en	actch_y_en	actch_x_en	actch_data_src	actch_npts<3:0>				0x00	RW
0x57	TAP_CONFIG				sel_axis<1:0>		tap_sensitivity<2:0>			0x00	RW
0x58	TAP_CONFIG1			quiet_dt<1:0>		quiet<1:0>		tics_th<1:0>		0x06	RW
0x7C	IF_CONF								spi3	0x00	RW
0x7D	SELF_TEST					acc_self_test_sign	acc_self_test_en_z	acc_self_test_en_y	acc_self_test_en_x	0x00	RW
0x7E	CMD	cmd<7:0>								0x00	RW

Register (0x00) CHIPID

DESCRIPTION: the register contains the chip identification code
read 0x90 to identify product
RESET: 0x90
DEFINITION (Go to register map):

Name	Register (0x00) CHIPID			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	1	0	0	1
Content	chipid_7_0			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	chipid_7_0			

Register (0x01) (reserved)

Register (0x02) ERR_REG

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

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

cmd_err: command (written to command register(0x7E)) execution failed. This is a clear-on-read bit.
cmd_err== 0x1: Command execution failed.

Register (0x03) STATUS

DESCRIPTION: the register contains the sensor status bits
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x03) STATUS**			
Bit	7	6	5	4
Read/Write	R	n/a	n/a	R
Reset Value	0	0	0	0
Content	drdy_stat	reserved		cmd_rdy
Bit	3	2	1	0
Read/Write	n/a	R	R	R
Reset Value	0	0	0	0
Content	reserved	power_mode_stat		int_active

int_active: the int_active bit is set if one of the interrupts is triggered

int_active		
0x00	not-triggered	one of the interrupts is triggered
0x01	triggered	one of the interrupts is triggered

power_mode_stat: current power mode of the sensor

power_mode_stat		
0x00	sleep_mode	device in sleep mode
0x01	low_power_mode	device in low power mode
0x02	normal_mode	device in normal mode

cmd_rdy: CMD (command register (0x7E)) decoder status.

cmd_rdy		
0x00	in_progress	command in progress
0x01	new_command	ready for new command

drdy_stat: data ready status is set as soon the accelerometer data conversion is ready

drdy_stat		
0x00	not-ready	data conversion not ready
0x01	ready	data conversion ready, new data available, clear on data read

Register (0x04) ACC_X_LSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x04) ACC_X_LSB**			
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			

acc_x_7_0: lsb of accelerometer x-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x05) ACC_X_MSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x05) ACC_X_MSB**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	acc_x_11_8			

acc_x_11_8: msb of accelerometer x-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x06) ACC_Y_LSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x06) ACC_Y_LSB**			
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			

acc_y_7_0: lsb of accelerometer y-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x07) ACC_Y_MSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x07) ACC_Y_MSB**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	acc_y_11_8			

acc_y_11_8: msb of accelerometer y-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x08) ACC_Z_LSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x08) ACC_Z_LSB**			
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			

acc_z_7_0: lsb of accelerometer z-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x09) ACC_Z_MSB

DESCRIPTION: Register for accelerometer data. The ACC_X_LSB-ACC_Z_MSB registers contain the latest data for x, y and z axis of accelerometer. A read operation on the register ACC_X_LSB-ACC_Z_MSB resets the INT_STAT0.drdy_int (data ready bit).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x09) ACC_Z_MSB**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	acc_z_11_8			

acc_z_11_8: msb of accelerometer z-axis data acceleration is obtained by the following operations:
acc_x/y/z = acc_x/y/z_7_0 + 256*acc_x/y/z_11_8
if(acc_x/y/z> 2047) acc_x/y/z=acc_x/y/z-4096

Register (0x0A) SENSOR_TIME0

DESCRIPTION: the register contains the sensor time
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0A) SENSOR_TIME0**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_7_0			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_7_0			

sensor_time_7_0: The internal sensor time is calculated using the formula
ttemp=sensor_time_7_0+256*sensor_time_15_8+65536*sensor_time_23_16
sensor_time=ttemp*312.5us

Register (0x0B) SENSOR_TIME1

DESCRIPTION: the register contains the sensor time
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0B) SENSOR_TIME1**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_15_8			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_15_8			

sensor_time_15_8: The internal sensor time is calculated using the formula
ttemp=sensor_time_7_0+256*sensor_time_15_8+65536*sensor_time_23_16
sensor_time=ttemp*312.5us

Register (0x0C) SENSOR_TIME2

DESCRIPTION: the register contains the sensor time
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0C) SENSOR_TIME2**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_23_16			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	sensor_time_23_16			

sensor_time_23_16: The internal sensor time is calculated using the formula
ttemp=sensor_time_7_0+256*sensor_time_15_8+65536*sensor_time_23_16
sensor_time=ttemp*312.5us

Register (0x0D) EVENT

DESCRIPTION: the register contains event bits.
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0D) EVENT**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	n/a	R
Reset Value	0	0	0	0
Content	reserved			por_detected

por_detected: power on reset bit , clear on read

por_detected		
0x00	no-por	no power up or softreset detected
0x01	por-detected	power up or softreset detected

Register (0x0E) INT_STAT0

DESCRIPTION: the registers contain the interrupt status bits
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0E) INT_STAT0**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	drdy_int_stat	fwm_int_stat	ffull_int_stat	ieng_overrun_stat
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	gen2_int_stat	gen1_int_stat	orientch_int_stat	wkup_int_stat

wkup_int_stat: wake-up interrupt status:
'0'= not set; '1'= set (wake-up condition is valid)

orientch_int_stat: orientation changed interrupt status:
'0'= not set; '1'= set (orientation is changed)

gen1_int_stat: generic interrupt 1 status:
'0'= not set; '1'= set

gen2_int_stat: generic interrupt 2 status:
'0'= not set; '1'= set

ieng_overrun_stat: issued when interrupt calculation could not be finished

ffull_int_stat: FIFO full interrupt status:
'0'= not set; '1'= set (FIFO full)

fwm_int_stat: FIFO watermark interrupt status:
'0'= not set; '1'= set

drdy_int_stat: data ready interrupt is status:
'0'= not set; '1'= set

Register (0x0F) INT_STAT1

DESCRIPTION: the registers contain the interrupt status bits
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x0F) INT_STAT1**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	R
Reset Value	0	0	0	0
Content	reserved			ieng_overrun_stat
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	d_tap_int_stat	s_tap_int_stat	step_int_stat	

step_int_stat: step detector interrupt status:
'0'= not set;
'1'= set (step detected);
'2'= set (step detected and very likely to have missed a step before; "double step");
'3'= not used

s_tap_int_stat: single tap interrupt status:
'0'= not set; '1'= set (single tap detected)

d_tap_int_stat: double tap interrupt status:
'0'= not set; '1'= set (double tap detected)

ieng_overrun_stat: issued when interrupt calculation could not be finished

Register (0x10) INT_STAT2

DESCRIPTION: the registers contain the interrupt status bits
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x10) INT_STAT2**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	R
Reset Value	0	0	0	0
Content	reserved			ieng_overrun_stat
Bit	3	2	1	0
Read/Write	n/a	R	R	R
Reset Value	0	0	0	0
Content	reserved	actch_z_int_stat	actch_y_int_stat	actch_x_int_stat

actch_x_int_stat: x-axis activity change detected: '0'= no change; '1'= changed

actch_y_int_stat: y-axis activity change detected: '0'= no change; '1'= changed

actch_z_int_stat: z-axis activity change detected: '0'= no change; '1'= changed

ieng_overrun_stat: issued when interrupt calculation could not be finished

Register (0x11) TEMP_DATA

DESCRIPTION: The register contains the temperature of the sensor.
The output word of the 8-bit temperature sensor is Two's complement.
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x11) TEMP_DATA			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	temp_data_7_0			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	temp_data_7_0			

temp_data_7_0: Conversion to real temperature is done using the formula
temp=((real)((signed)temp_data))*0.5+23.0

Register (0x12) FIFO_LENGTH0

DESCRIPTION: the register contains the number of bytes stored in FIFO
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x12) FIFO_LENGTH0			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	fifo_bytes_cnt_7_0			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	fifo_bytes_cnt_7_0			

fifo_bytes_cnt_7_0: This is the LSBs data of the FIFO size count
fifo_size_bytes = fifo_bytes_cnt_7_0 + 256*fifo_bytes_cnt_10_8

Register (0x13) FIFO_LENGTH1

DESCRIPTION: the register contains the number of bytes stored in FIFO
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x13) FIFO_LENGTH1**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	R	R	R
Reset Value	0	0	0	0
Content	reserved	fifo_bytes_cnt_10_8		

fifo_bytes_cnt_10_8: This is the MSBs data of the FIFO size count
fifo_size_bytes = fifo_bytes_cnt_7_0 + 256*fifo_bytes_cnt_10_8

Register (0x14) FIFO_DATA

DESCRIPTION: the register contains the FIFO data. The FIFO data can be read out as burst read. The number of bytes written in the FIFO to be read is stored in the registers FIFO_LENGTH(0/1).
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x14) FIFO_DATA**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	fifo_data_field			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	fifo_data_field			

Register (0x15) STEP_CNT_0

DESCRIPTION: the register contains the number of steps detected by step counter.
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x15) STEP_CNT_0**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_7_0			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_7_0			

step_cnt_7_0: step_count=step_cnt_7_0+256*step_cnt_15_8+65536*step_cnt_23_16

Register (0x16) STEP_CNT_1

DESCRIPTION: the register contains the number of steps detected by step counter.
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x16) STEP_CNT_1**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_15_8			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_15_8			

step_cnt_15_8: step_count=step_cnt_7_0+256*step_cnt_15_8+65536*step_cnt_23_16

Register (0x17) STEP_CNT_2

DESCRIPTION: the register contains the number of steps detected by step counter.
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x17) STEP_CNT_2**			
Bit	7	6	5	4
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_23_16			
Bit	3	2	1	0
Read/Write	R	R	R	R
Reset Value	0	0	0	0
Content	step_cnt_23_16			

step_cnt_23_16: step_count=step_cnt_7_0+256*step_cnt_15_8+65536*step_cnt_23_16

Register (0x18) STEP_STAT

DESCRIPTION: the register filed contains the status STILL(00), WALK(01) or RUN(01)
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x18) STEP_STAT**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	R	R
Reset Value	0	0	0	0
Content	reserved		step_stat_field	

step_stat_field		
0x00	still	no walking, no running
0x01	walking	step counter detects walking activity
0x02	running	step counter detects running activity

Register (0x19) ACC_CONFIG0

DESCRIPTION: the registers contain the accelerometer configuration.
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x19) ACC_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	n/a
Reset Value	0	0	0	0
Content	filt1_bw	osr_lp		reserved
Bit	3	2	1	0
Read/Write	n/a	n/a	RW	RW
Reset Value	0	0	0	0
Content	reserved		power_mode_conf	

power_mode_conf: '000'= sleep; '001'= low power mode; '010'= normal mode

power_mode_conf		
0x00	sleep_mode	sleep mode
0x01	low_power_mode	low power mode
0x02	normal_mode	normal mode
0x03	reserved	switches to sleep mode (0x0)

osr_lp: oversampling ratio for low power mode

filt1_bw		
0x00	high	0.4x ODR
0x01	low	0.2x ODR

Register (0x1A) ACC_CONFIG1

DESCRIPTION: the registers contain the accelerometer configuration
RESET: 0x49
DEFINITION (Go to register map):

Name	**Register (0x1A) ACC_CONFIG1**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	1	0	0
Content	acc_range		osr	
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	1	0	0	1
Content	acc_odr			

acc_odr: output data rate of accelerometer for acc_filt1

acc_odr	**ODR [Hz]**
0x00	reserved
0x01	reserved
0x02	reserved
0x03	reserved
0x04	reserved
0x05	12.5
0x06	25
0x07	50
0x08	100
0x09	200
0x0a	400
0x0b	800
0x0c	reserved
0x0d	reserved
0x0e	reserved
0x0f	reserved

osr: oversampling rate 0/1/2/3 for normal mode
osr=0: lowest power, lowest oversampling rate, lowest accuracy
osr=3: highest accuracy, higest oversampling rate, highest power
settings 0, 1, 2 and 3 allow linearly trading power versus accuracy(noise)

acc_range: accelerometer measurement range

acc_range		
0x00	2g	+/-2g measurement range
0x01	4g	+/-4g measurement range
0x02	8g	+/-8g measurement range
0x03	16g	+/-16g measurement range

Register (0x1B) ACC_CONFIG2

DESCRIPTION: the registers contain the accelerometer configuration
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x1B) ACC_CONFIG2**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	n/a	n/a
Reset Value	0	0	0	0
Content	data_src_reg		reserved	

data_src_reg: Select source for data registers

data_src_reg		
0x00	acc_filt1	variable ODR filter
0x01	acc_filt2	fixed 100Hz output data rate filter
0x02	acc_filt_lp	fixed 100Hz output data rate filter, 1Hz bandwidth
0x03	acc_filt1	variable ODR filter

Register (0x1F) INT_CONFIG0

DESCRIPTION: The register contains interrupt control bits, 0 = not enabled, 1 = enabled
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x1F) INT_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	n/a
Reset Value	0	0	0	0
Content	drdy_int_en	fwm_int_en	ffull_int_en	reserved
Bit	3	2	1	0
Read/Write	RW	RW	RW	n/a
Reset Value	0	0	0	0
Content	gen2_int_en	gen1_int_en	orientch_int_en	reserved

orientch_int_en: orientation changed interrupt
gen1_int_en: generic interrupt 1
gen2_int_en: generic interrupt 2
ffull_int_en: FIFO full interrupt
fwm_int_en: FIFO watermark interrupt
drdy_int_en: data ready interrupt

Register (0x20) INT_CONFIG1

DESCRIPTION: The register contains interrupt control bits, 0 = not enabled, 1 = enabled
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x20) INT_CONFIG1**			
Bit	7	6	5	4
Read/Write	RW	n/a	n/a	RW
Reset Value	0	0	0	0
Content	latch_int	reserved		actch_int_en
Bit	3	2	1	0
Read/Write	RW	RW	n/a	RW
Reset Value	0	0	0	0
Content	d_tap_int_en	s_tap_int_en	reserved	step_int_en

step_int_en: step detected interrupt (step counter)
s_tap_int_en: single tap interrupt
d_tap_int_en: double tap interrupt
actch_int_en: activity changed interrupt
latch_int: latched interrupt mode configuration

latch_int		
0x00	nolatch	non-latched mode
0x01	latching	latching mode

Register (0x21) INT1_MAP

DESCRIPTION: The register contains the interrupt to physical INT1 pin mapping
0: interrupt is not mapped to INT1
1: interrupt is mapped to pin INT1
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x21) INT1_MAP**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	drdy_int1	fwm_int1	ffull_int1	ieng_overrun_int1
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int1	gen1_int1	orientch_int1	wkup_int1

wkup_int1: wake-up interrupt is mapped to int1
orientch_int1: orientation changed interrupt is mapped to int1
gen1_int1: generic interrupt 1 is mapped to int1
gen2_int1: generic interrupt 2 is mapped to int1

ieng_overrun_int1: interrupt overrun mapped to int1
ffull_int1: fifo full interrupt is mapped to int1
fwm_int1: fifo watermark interrupt is mapped to int1
drdy_int1: data ready interrupt is mapped to int1

Register (0x22) INT2_MAP

DESCRIPTION: The register contains the interrupt to physical INT2 pin mapping
0: interrupt is not mapped to INT2
1: interrupt is mapped to pin INT2
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x22) INT2_MAP			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	drdy_int2	fwm_int2	ffull_int2	ieng_overrun_int2
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int2	gen1_int2	orientch_int2	wkup_int2

wkup_int2: wake-up interrupt is mapped to INT2
orientch_int2: orientation changed interrupt is mapped to INT2
gen1_int2: generic interrupt 1 is mapped to INT2
gen2_int2: generic interrupt 2 is mapped to INT2
ieng_overrun_int2: interrupt overrun mapped to int2
ffull_int2: fifo full interrupt is mapped to INT2
fwm_int2: fifo watermark interrupt is mapped to INT2
drdy_int2: data ready interrupt is mapped to INT2

Register (0x23) INT12_MAP

DESCRIPTION: the registers contain the interrupts mapping to physical pins
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x23) INT12_MAP			
Bit	7	6	5	4
Read/Write	RW	RW	n/a	RW
Reset Value	0	0	0	0
Content	actch_int2	tap_int2	reserved	step_int2
Bit	3	2	1	0
Read/Write	RW	RW	n/a	RW
Reset Value	0	0	0	0
Content	actch_int1	tap_int1	reserved	step_int1

step_int1: step detector interrupt is mapped to INT1

tap_int1: tap sensing interrupt is mapped to INT1
actch_int1: activity changed interrupt is mapped to INT1
step_int2: step detector interrupt is mapped to INT2
tap_int2: tap sensing interrupt is mapped to INT2
actch_int2: activity changed interrupt is mapped to INT2

Register (0x24) INT12_IO_CTRL

DESCRIPTION: the register contains physical behaviour of interrupt pins configurations
RESET: 0x22
DEFINITION (Go to register map):

Name	**Register (0x24) INT12_IO_CTRL**			
Bit	7	6	5	4
Read/Write	n/a	RW	RW	n/a
Reset Value	0	0	1	0
Content	reserved	int2_od	int2_lvl	reserved
Bit	3	2	1	0
Read/Write	n/a	RW	RW	n/a
Reset Value	0	0	1	0
Content	reserved	int1_od	int1_lvl	reserved

int1_lvl: INT1 pin output level

int1_lvl	
0x00	interrupt pin INT1 low-active
0x01	interrupt pin INT1 high-active

int1_od: INT1 pin output driver mode: CMOS or open drain

int1_od		
0x00	Push-pull	CMOS push-pull drive characteristic
0x01	open drain	

int2_lvl: INT2 pin output level

int2_lvl	
0x00	interrupt pin INT2 low-active
0x01	interrupt pin INT2 high-active

int2_od: INT2 pin output driver mode: see interrupt physical behaviour

int2_od		
0x00	Push-pull	CMOS push-pull drive characteristic
0x01	open drain	

Register (0x26) FIFO_CONFIG0

DESCRIPTION: the registers contain the FIFO control and FIFO configuration settings
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x26) FIFO_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	fifo_z_en	fifo_y_en	fifo_x_en	fifo_8bit_en
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	fifo_data_src	fifo_time_en	fifo_stop_on_full	auto_flush

auto_flush: auto flush FIFO when changing power-mode

auto_flush		
0x00	noaction	no FIFO flush on changing power mode
0x01	fifo-flush	FIFO flush on changing power mode

fifo_stop_on_full: FIFO writing - stream mode / FIFO full mode

fifo_stop_on_full		
0x00	streaming	overwrite oldest FIFO data when FIFO full
0x01	fifo-stop-on-full	stop writing into FIFO when full

fifo_time_en: Enable sending of sensortime frame when reading burst from FIFO and the FIFO runs empty

fifo_time_en	
0x00	Disable sensortime in FIFO
0x01	Enable sensortime in FIFO

fifo_data_src: acceleration data source for storing into FIFO

fifo_data_src		
0x00	acc_filt1	store data from acc_filt1 (variable data rate) in FIFO
0x01	acc_filt2	store data from acc_filt2 (100Hz data rate) in FIFO

fifo_8bit_en: enables 8 bit FIFO mode

fifo_8bit_en		
0x00	12bit	store data in 12bit format (default), two bytes occupied
0x01	8bit	store data in 8bit format

fifo_x_en: x-channel data storage control

fifo_x_en		
0x00	nostore	do not store x axis data
0x01	store	store x axis data

fifo_y_en: y-channel data storage control

fifo_y_en		
0x00	nostore	do not store y axis data
0x01	store	store y axis data

fifo_z_en: z-channel data storage control

fifo_z_en		
0x00	nostore	do not store z axis data
0x01	store	store z axis data

Register (0x27) FIFO_CONFIG1

DESCRIPTION: the registers contain the FIFO control and FIFO configuration settings
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x27) FIFO_CONFIG1**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	fifo_watermark_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	fifo_watermark_7_0			

fifo_watermark_7_0: LSB of FIFO watermark threshold configuration:
watermark[byte]= fifo_watermark_7_0 + 256*fifo_watermark_10_8

Register (0x28) FIFO_CONFIG2

DESCRIPTION: the registers contain the FIFO control and FIFO configuration settings
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x28) FIFO_CONFIG2**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	RW	RW	RW
Reset Value	0	0	0	0
Content	reserved	fifo_watermark_10_8		

fifo_watermark_10_8: MSB of FIFO watermark threshold configuration
watermark[byte]= fifo_watermark_7_0 + 256*fifo_watermark_10_8

Register (0x29) FIFO_PWR_CONFIG

DESCRIPTION: the registers contain the FIFO read power circuit settings, saves 100nA when set
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x29) FIFO_PWR_CONFIG**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	n/a	RW
Reset Value	0	0	0	0
Content	reserved			fifo_read_disable

fifo_read_disable: manual disable for the FIFO read power circuit when set HIGH

Register (0x2A) AUTOLOWPOW_0

DESCRIPTION: the registers contain configurations for auto-low-power condition
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x2A) AUTOLOWPOW_0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	auto_lp_timeout_thres_11_4			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	auto_lp_timeout_thres_11_4			

auto_lp_timeout_thres_11_4: MSB of auto-low-power timeout threshold
temp = auto_lp_timeout_thres_3_0 +16*auto_lp_timeout_thres_11_4
timeout= temp*2.5ms

Register (0x2B) AUTOLOWPOW_1

DESCRIPTION: the registers contain configurations for auto-low-power condition
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x2B) AUTOLOWPOW_1**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	auto_lp_timeout_thres_3_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	auto_lp_timeout		gen1_int	drdy_lowpow_trig

drdy_lowpow_trig: data ready as source for auto-low-power condition

drdy_lowpow_trig		
0x00	notrig	no triggering of low-power
0x01	trig-newdata	new data ready triggers going into low-power

gen1_int: generic interrupt 1 as source for auto-low-power condition

gen1_int		
0x00	nodtrig	no triggering of low-power
0x01	trig-gen1	generic interrupt 1 triggers going into low-power

auto_lp_timeout: auto-low-power timeout as source for auto-low-power condition

auto_lp_timeout		
0x00	auto_lp_timeout_0	Low-power timeout disabled
0x01	auto_lp_timeout_1	Low-power timeout active, device shall switch into low power mode as soon timeout counter is expired
0x02	auto_lp_timeout_2	Low-power timeout active, as 0x01, but timeout counter resets if gen2_int is asserted
0x03	auto_lp_timeout_3	same as 0x01

auto_lp_timeout_thres_3_0: LSB of auto-low-power timeout threshold
temp = auto_lp_timeout_thres_3_0 +16*auto_lp_timeout_thres_11_4
timeout= temp*2.5ms

Register (0x2C) AUTOWAKEUP_0

DESCRIPTION: the register contains configurations for auto-wake-up condition.
The auto-wake-up condition is evaluated as soon as the sensor changes into low power mode
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x2C) AUTOWAKEUP_0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	wakeup_timeout_thres_11_4			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	wakeup_timeout_thres_11_4			

wakeup_timeout_thres_11_4: MSB of wake-up timeout threshold
temp= wakeup_timeout_thres_3_0 + 16*wakeup_timeout_thres_11_4
timeout=temp=2.5ms

Register (0x2D) AUTOWAKEUP_1

DESCRIPTION: the register contains configurations for auto-wake-up condition.
The auto-wake-up condition is evaluated as soon as the sensor changes into low power mode
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x2D) AUTOWAKEUP_1**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	wakeup_timeout_thres_3_0			
Bit	3	2	1	0
Read/Write	n/a	RW	RW	n/a
Reset Value	0	0	0	0
Content	reserved	wkup_timeout	wkup_int	reserved

wkup_int: wake-up interrupt as source for auto-wake-up condition

wkup_int		
0x00	nowakeup	not use wake-up interrupt for auto-wake-up
0x01	en-wakeup	use wake-up interrupt for auto-wake-up

wkup_timeout: wake-up timeout as source for auto-wake-up condition

wkup_timeout		
0x00	no-timeout	timer not used for auto-wake-up
0x01	enab-timeout	timer triggers auto-wake-up

wakeup_timeout_thres_3_0: LSB of wake-up timeout threshold
temp= wakeup_timeout_thres_3_0 + 16*wakeup_timeout_thres_11_4
timeout=temp*2.5ms

Register (0x2F) WKUP_INT_CONFIG0

DESCRIPTION: the registers contain configurations for wake-up interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x2F) WKUP_INT_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	wkup_z_en	wkup_y_en	wkup_x_en	num_of_samples
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	num_of_samples		wkup_refu	

wkup_refu: wake-up interrupt reference update mode

wkup_refu		
0x00	manual	manual update (reference registers are updated by external MCU)
0x01	onetime	one time automated update before going into low power mode
0x02	everytime	every time after data conversion

num_of_samples: number of data samples used for interrupt condition evaluation, allowed range 0...7 (number of samples(real) = num_of_samples(set) + 1)

wkup_x_en: enable wake-up interrupt for x channel

wkup_x_en		
0x00	disabled	no x axis evaluation
0x01	enabled	wakeup function evaluates x axis

wkup_y_en: enable wake-up interrupt for y channel

wkup_y_en		
0x00	disabled	no y axis evaluation
0x01	enabled	wakeup function evaluates y axis

wkup_z_en: enable wake-up interrupt for z channel

wkup_z_en		
0x00	disabled	no z axis evaluation
0x01	enabled	wakeup function evaluates z axis

Register (0x30) WKUP_INT_CONFIG1

DESCRIPTION: the register contains configurations for wake-up interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x30) WKUP_INT_CONFIG1			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_thres			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_thres			

int_wkup_thres: interrupt threshold
The value is range sensitive, unsigned and defines the threshold for activity which must be present to cause a wake-up. The physical value of LSB corresponds to 2^(2+acc_range)/256, here acc_range = 0, 1, 2, 3 corresponds to +/-2g, +/-4g, +/-8g, +/-16g.

Register (0x31) WKUP_INT_CONFIG2

DESCRIPTION: the register contains configurations for wake-up interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x31) WKUP_INT_CONFIG2			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refx			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refx			

int_wkup_refx: reference acceleration x-axis for the wake-up interrupt. The value is range sensitive and a signed integer, either provided by the host (wkup_refu=0) or automatically the wake-up interrupt calculates abs(acc_x/y/z-int_wkup_refx/y/z*16)>int_wkup_thres*16 to determine whether activity is sufficiently high on the x/y/z-axis to cause wake-up

Register (0x32) WKUP_INT_CONFIG3

DESCRIPTION: the register contains configurations for wake-up interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x32) WKUP_INT_CONFIG3**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refy			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refy			

int_wkup_refy: reference acceleration y-axis for the wake-up interrupt. The value is range sensitive and a signed integer, either provided by the host (wkup_refu=0) or automatically the wake-up interrupt calculates abs(acc_x/y/z-int_wkup_refx/y/z*16)>int_wkup_thres*16 to determine whether activity is sufficiently high on the x/y/z-axis to cause wake-up

Register (0x33) WKUP_INT_CONFIG4

DESCRIPTION: the register contains configurations for wake-up interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x33) WKUP_INT_CONFIG4**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refz			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_wkup_refz			

int_wkup_refz: reference acceleration z-axis for the wake-up interrupt. The value is range sensitive and a signed integer, either provided by the host (wkup_refu=0) or automatically the wake-up interrupt calculates abs(acc_x/y/z-int_wkup_refx/y/z*16)>int_wkup_thres*16 to determine whether activity is sufficiently high on the x/y/z-axis to cause wake-up

Register (0x35) ORIENTCH_CONFIG0

DESCRIPTION: the registers contain configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x35) ORIENTCH_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_z_en	orient_y_en	orient_x_en	orient_data_src
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_refu		00	

Note: Bit 0 and 1 must be written with a 0x0

orient_refu: reference update mode for orientation changed interrupt

orient_refu		
0x00	manual	manual update (reference registers are updated by serial interface command)
0x01	onetime_2	one time automated update using acc_filt2 data
0x02	onetime_lp	one time automated update using acc_filt_lp data

orient_data_src: data source selection for orientation changed interrupt evaluation

orient_data_src		
0x00	filt2	data source is acc_filt2
0x01	filt_lp	data source is acc_filt_lp

orient_x_en: enable orientation changed interrupt for x-axis: 0-not active;1-active
orient_y_en: enable orientation changed interrupt for y-axis: 0-not active;1-active
orient_z_en: enable orientation changed interrupt for z-axis: 0-not active;1-active

Register (0x36) ORIENTCH_CONFIG1

DESCRIPTION: the registers contain configurations for orientation change interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x36) ORIENTCH_CONFIG1			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_thres			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_thres			

orient_thres: threshold configuration for orientation changed interrupt 8mg/lsb resolution

Register (0x38) ORIENTCH_CONFIG3

DESCRIPTION: the registers contain configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x38) ORIENTCH_CONFIG3			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_dur			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	orient_dur			

orient_dur: duration for (stable) new orientation before interrupt is triggered
duration is a multiple of the number of data samples processed (ODR=100HZ) from the selected filter

Register (0x39) ORIENTCH_CONFIG4

DESCRIPTION: the register contains configurations for orientation change interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x39) ORIENTCH_CONFIG4**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refx_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refx_7_0			

int_orient_refx_7_0: LSB of x-axis reference for orientation change evaluation
the value is a signed integer, either provided by the host (orient_refu=0) or automatically the interrupt calculates
abs(acc_x/y/z-int_orientch_refx/y/z)>orient_thres to determine whether activity is sufficiently high on the x/y/z-axis to cause an interrupt trigger

int_orientch_refx = int_orient_refx_7_0 + 256*int_orient_refx_11_8

Register (0x3A) ORIENTCH_CONFIG5

DESCRIPTION: the register contains configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x3A) ORIENTCH_CONFIG5			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refx_11_8			

int_orient_refx_11_8: MSB of x-axis reference for orientation change evaluation. The value is range sensitive

Register (0x3B) ORIENTCH_CONFIG6

DESCRIPTION: the register contains configurations for orientation change interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x3B) ORIENTCH_CONFIG6			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refy_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refy_7_0			

int_orient_refy_7_0: LSB of y-axis reference for orientation change evaluation
The value is range sensitive and a signed integer, either provided by the host (orient_refu=0) or automatically the interrupt calculates abs(acc_x/y/z-int_orientch_refx/y/z)>orient_thres to determine whether activity is sufficiently high on the x/y/z-axis to cause an interrupt trigger int_orientch_refy = int_orient_refy_7_0 + 256*int_orient_refy_11_8

Register (0x3C) ORIENTCH_CONFIG7

DESCRIPTION: the register contains configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x3C) ORIENTCH_CONFIG7**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refy_11_8			

int_orient_refy_11_8: MSB of y-axis reference for orientation change evaluation

Register (0x3D) ORIENTCH_CONFIG8

DESCRIPTION: the registers contain configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x3D) ORIENTCH_CONFIG8**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refz_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refz_7_0			

int_orient_refz_7_0: LSB of z-axis reference for orientation change evaluation
The value is range sensitive and a signed integer, either provided by the host (orient_refu=0) or automatically the interrupt calculates abs(acc_x/y/z-int_orientch_refx/y/z)>orient_thres to determine whether activity is sufficiently high on the x/y/z-axis to cause an interrupt trigger int_orientch_refz = int_orient_refz_7_0 + 256*int_orient_refz_11_8

Register (0x3E) ORIENTCH_CONFIG9

DESCRIPTION: the registers contain configurations for orientation changed interrupt
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x3E) ORIENTCH_CONFIG9**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	int_orient_refz_11_8			

int_orient_refz_11_8: MSB of z-axis reference for orientation change evaluation

Register (0x3F) GEN1INT_CONFIG0

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x3F) GEN1INT_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_act_z_en	gen1_act_y_en	gen1_act_x_en	gen1_data_src
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_act_refu		gen1_act_hyst	

gen1_act_hyst: hysteresis configuration for interrupt evaluation

gen1_act_hyst		
0x00	not-active	no hysteresis
0x01	24mg	24mg hysteresis
0x02	48mg	48mg hysteresis
0x03	96mg	96mg hysteresis

gen1_act_refu: reference update mode for evaluation

gen1_act_refu		
0x00	manual	manual update (reference registers are updated by a serial interface command)
0x01	onetime	one time automated update by the selected data source
0x02	everytime	every time automated update by the selected data source
0x03	everytime_lp	every time automated update by acc_filt_lp

gen1_data_src: data source selection for interrupts evaluation

gen1_data_src		
0x00	filt1	data source is acc_filt1
0x01	filt2	data source is acc_filt2

gen1_act_x_en: x-axis channel control for interrupt evaluation: '0'- not active; '1'- active
gen1_act_y_en: y-axis channel control for interrupt evaluation: '0'- not active; '1'- active
gen1_act_z_en: z-axis channel control for interrupt evaluation: '0'- not active; '1'- active

Register (0x40) GEN1INT_CONFIG1

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x40) GEN1INT_CONFIG1**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	RW	RW
Reset Value	0	0	0	0
Content	reserved		gen1_criterion_sel	gen1_comb_sel

gen1_comb_sel: Select logical combination for creating the interrupt signal from the individual axes that have been enabled

gen1_comb_sel		
0x00	OR	OR combination of x/y/z axis evaluation results
0x01	AND	AND combination of x/y/z axis evaluation results

gen1_criterion_sel: Select criterion for threshold comparison

gen1_criterion_sel		
0x00	inactivity	acceleration below threshold: inactivity detection
0x01	activity	acceleration above threshold: activity detection

Register (0x41) GEN1INT_CONFIG2

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x41) GEN1INT_CONFIG2**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_thres			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_thres			

gen1_int_thres: threshold configuration for detection: 8 mg/lsb
unsigned integer

Register (0x42) GEN1INT_CONFIG3

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x42) GEN1INT_CONFIG3**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_dur_15_8			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_dur_15_8			

gen1_int_dur_15_8: duration for which the condition has to persist until interrupt can be triggered
duration is measured in data samples of selected data source
gen1_int_dur= 256*gen1_int_dur_15_8 + gen1_int_dur_7_0

Register (0x43) GEN1INT_CONFIG31

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x43) GEN1INT_CONFIG31**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_dur_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_dur_7_0			

gen1_int_dur_7_0: duration for which the condition has to persist until interrupt can be triggered
duration is measured in data samples of selected data source
gen1_int_dur= 256*gen1_int_dur_15_8 + gen1_int_dur_7_0

Register (0x44) GEN1INT_CONFIG4

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x44) GEN1INT_CONFIG4**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refx_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refx_7_0			

gen1_int_th_refx_7_0: LSB of reference x-axis value for evaluation. The value is range sensitive.
gen1_int_refx = gen1_int_th_refx_7_0 + 256*gen1_int_th_refx_11_8

Register (0x45) GEN1INT_CONFIG5

DESCRIPTION: the register contains configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x45) GEN1INT_CONFIG5			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refx_11_8			

gen1_int_th_refx_11_8: MSB of reference x-axis value for evaluation. The value is range sensitive.
gen1_int_refx = gen1_int_th_refx_7_0 + 256*gen1_int_th_refx_11_8

Register (0x46) GEN1INT_CONFIG6

DESCRIPTION: the register contains configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x46) GEN1INT_CONFIG6			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refy_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refy_7_0			

gen1_int_th_refy_7_0: LSB of reference y-axis value for evaluation. The value is range sensitive.
gen1_int_refy = gen1_int_th_refy_7_0 + 256*gen1_int_th_refy_11_8

Register (0x47) GEN1INT_CONFIG7

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x47) GEN1INT_CONFIG7**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refy_11_8			

gen1_int_th_refy_11_8: MSB of reference y-axis value for evaluation. The value is range sensitive.
gen1_int_refy = gen1_int_th_refy_7_0 + 256*gen1_int_th_refx_11_8

Register (0x48) GEN1INT_CONFIG8

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x48) GEN1INT_CONFIG8**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refz_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refz_7_0			

gen1_int_th_refz_7_0: LSB of reference z-axis value for evaluation. The value is range sensitive.
gen1_int_refz = gen1_int_th_refz_7_0 + 256*gen1_int_th_refz_11_8

Register (0x49) GEN1INT_CONFIG9

DESCRIPTION: the registers contain configurations for generic interrupt 1 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x49) GEN1INT_CONFIG9**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen1_int_th_refz_11_8			

gen1_int_th_refz_11_8: MSB of reference z-axis value for evaluation. The value is range sensitive.

gen1_int_refz = gen1_int_th_refz_7_0 + 256*gen1_int_th_refz_11_8

Register (0x4A) GEN2INT_CONFIG0

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x4A) GEN2INT_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_act_z_en	gen2_act_y_en	gen2_act_x_en	gen2_data_src
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_act_refu		gen2_act_hyst	

gen2_act_hyst: hysteresis configuration for interrupt evaluation

gen2_act_hyst		
0x00	not-active	no hysteresis
0x01	24mg	24mg hysteresis
0x02	48mg	48mg hysteresis
0x03	96mg	96mg hysteresis

gen2_act_refu: reference update mode for evaluation

gen2_act_refu		
0x00	manual	manual update (reference registers are updated by serial interface command)
0x01	onetime	one time automated update by the selected data source
0x02	everytime	every time automated update by the selected data source
0x03	everytime_lp	every time automated update by acc_filt_lp

gen2_data_src: data source selection for interrupts evaluation

gen2_data_src		
0x00	filt1	data source is acc_filt1
0x01	filt2	data source is acc_filt2

gen2_act_x_en: x-axis channel control for interrupt evaluation: 0 - not active; 1 - active
gen2_act_y_en: y-axis channel control for interrupt evaluation: 0 - not active; 1 - active
gen2_act_z_en: z-axis channel control for interrupt evaluation: 0 - not active; 1 - active

Register (0x4B) GEN2INT_CONFIG1

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x4B) GEN2INT_CONFIG1**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	RW	RW
Reset Value	0	0	0	0
Content	reserved		gen2_criterion_sel	gen2_comb_sel

gen2_comb_sel: Select logical combination for creating the interrupt signal from the individual axes that have been enabled

gen2_comb_sel		
0x00	OR	OR combination of x/y/z axis evaluation results
0x01	AND	AND combination of x/y/z axis evaluation results

gen2_criterion_sel: Select criterion for threshold comparison

gen2_criterion_sel		
0x00	inactivity	acceleration below threshold: inactivity detection
0x01	activity	acceleration above threshold: activity detection

Register (0x4C) GEN2INT_CONFIG2

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x4C) GEN2INT_CONFIG2			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_thres			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_thres			

gen2_int_thres: threshold configuration for interrupt detection: 8 mg/LSB
unsigned integer

Register (0x4D) GEN2INT_CONFIG3

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x4D) GEN2INT_CONFIG3			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_dur_15_8			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_dur_15_8			

gen2_int_dur_15_8: duration for which the condition has to persist until interrupt can be triggered
duration is measured in data samples of selected data source
gen2_int_dur= 256*gen2_int_dur_15_8 + gen2_int_dur_7_0

Register (0x4E) GEN2INT_CONFIG31

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x4E) GEN2INT_CONFIG31			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_dur_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_dur_7_0			

gen2_int_dur_7_0: duration for which the condition has to persist until interrupt can be triggered
duration is measured in data samples of selected data source
gen2_int_dur= 256*gen2_int_dur_15_8 + gen2_int_dur_7_0

Register (0x4F) GEN2INT_CONFIG4

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x4F) GEN2INT_CONFIG4			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refx_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refx_7_0			

gen2_int_th_refx_7_0: LSB of reference x-axis value for evaluation. The value is range sensitive.
gen2_int_refx = gen2_int_th_refx_7_0 + 256*gen2_int_th_refx_11_8

Register (0x50) GEN2INT_CONFIG5

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x50) GEN2INT_CONFIG5			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refx_11_8			

gen2_int_th_refx_11_8: MSB of reference x-axis value for evaluation. The value is range sensitive.
gen2_int_refx = gen2_int_th_refx_7_0 + 256*gen2_int_th_refx_11_8

Register (0x51) GEN2INT_CONFIG6

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	Register (0x51) GEN2INT_CONFIG6			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refy_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refy_7_0			

gen2_int_th_refy_7_0: LSB of reference y-axis value for evaluation. The value is range sensitive.
gen2_int_refy = gen2_int_th_refy_7_0 + 256*gen2_int_th_refy_11_8

Register (0x52) GEN2INT_CONFIG7

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x52) GEN2INT_CONFIG7**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refy_11_8			

gen2_int_th_refy_11_8: MSB of reference y-axis value for evaluation. The value is range sensitive.
gen2_int_refy = gen2_int_th_refy_7_0 + 256*gen2_int_th_refy_11_8

Register (0x53) GEN2INT_CONFIG8

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x53) GEN2INT_CONFIG8**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refz_7_0			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refz_7_0			

gen2_int_th_refz_7_0: LSB of reference z-axis value for evaluation. The value is range sensitive.
gen2_int_refz = gen2_int_th_refz_7_0 + 256*gen2_int_th_refz_11_8

Register (0x54) GEN2INT_CONFIG9

DESCRIPTION: the registers contain configurations for generic interrupt 2 evaluation
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x54) GEN2INT_CONFIG9**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	gen2_int_th_refz_11_8			

gen2_int_th_refz_11_8: MSB of reference z-axis value for evaluation. The value is range sensitive.
gen2_int_refz = gen2_int_th_refz_7_0 + 256*gen2_int_th_refz_11_8

Register (0x55) ACTCH_CONFIG0

DESCRIPTION: Activity changed interrupt configuration registers
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x55) ACTCH_CONFIG0**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	actch_thres			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	actch_thres			

actch_thres: threshold configuration for activity changed interrupt: 8mg/g resolution

Register (0x56) ACTCH_CONFIG1

DESCRIPTION: Activity changed interrupt configuration registers
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x56) ACTCH_CONFIG1**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	actch_z_en	actch_y_en	actch_x_en	actch_data_src
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	actch_npts			

actch_npts: number of points for evaluation of the activity: 32, 64, 128, 256, 512

actch_npts		
0x00	32	32 points
0x01	64	64 points
0x02	128	128 points
0x03	256	256 points
0x04	512	512 points
0x05-0x0F		reserved

actch_data_src: data source

actch_data_src		
0x00	actch_use_acc_filt1	
0x01	actch_use_acc_filt2	

actch_x_en: activity changed evaluation for x-axis enabled: '0'- not active; '1'- active
actch_y_en: activity changed evaluation for y-axis enabled: '0'- not active; '1'- active
actch_z_en: activity changed evaluation for z-axis enabled: '0'- not active; '1'- active

Register (0x57) TAP_CONFIG

DESCRIPTION: tap interrupt configuration registers
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x57) TAP_CONFIG**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	RW
Reset Value	0	0	0	0
Content	reserved			sel_axis
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	sel_axis	tap_sensitivity		

tap_sensitivity: sensitivity of the tap algorithm
0: highest sensitivity
7: lowest sensitivity
possible values range from 0 to 7 for a monotonic decrease of sensitivity with every setting

sel_axis: Modifies the selection of the data provided to the algorithm

sel_axis		
0x00	Z	use Z axis data
0x01	Y	use Y axis data
0x02	X	use X axis data

Register (0x58) TAP_CONFIG1

DESCRIPTION: tap interrupt configuration registers
RESET: 0x06
DEFINITION (Go to register map):

Name	**Register (0x58) TAP_CONFIG1**			
Bit	7	6	5	4
Read/Write	n/a	n/a	RW	RW
Reset Value	0	0	0	0
Content	reserved		quiet_dt	
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	1	1	0
Content	quiet		tics_th	

tics_th: Maximum time between upper and lower peak of a tap, in data samples
this time depends on the mechanics of the device tapped onto
default = 12 samples

tics_th		
0x00	6	6 data samples for high-low tap signal change time
0x01	9	9 data samples for high-low tap signal change time
0x02	12	12 data samples for high-low tap signal change time
0x03	18	18 data samples for high-low tap signal change time

quiet: Minimum quiet time before and after double tap, in data samples
This time also defines the longest time interval between two taps so that they are considered as double tap

quiet		
0x00	60	60 data samples quiet tie between single or doube taps
0x01	80	80 data samples quiet tie between single or doube taps
0x02	100	100 data samples quiet tie between single or doube taps
0x03	120	120 data samples quiet tie between single or doube taps

quiet_dt: Minimum time between the two taps of a double tap, in data samples

quiet_dt		
0x00	4	4 data samples minimum time between double taps
0x01	8	8 data samples minimum time between double taps
0x02	12	12 data samples minimum time between double taps
0x03	16	16 data samples minimum time between double taps

Register (0x7C) IF_CONF

DESCRIPTION: Serial interface settings
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x7C) IF_CONF**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	n/a	n/a	n/a	RW
Reset Value	0	0	0	0
Content	reserved			spi3

spi3: Configure SPI Interface Mode for primary interface

spi3		
0x00	spi4	SPI 4-wire mode
0x01	spi3	SPI 3-wire mode

Register (0x7D) SELF_TEST

DESCRIPTION: Settings for the sensor self-test configuration and trigger
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x7D) SELF_TEST**			
Bit	7	6	5	4
Read/Write	n/a	n/a	n/a	n/a
Reset Value	0	0	0	0
Content	reserved			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	acc_self_test_sign	acc_self_test_en_z	acc_self_test_en_y	acc_self_test_en_x

acc_self_test_en_x: trigger self test for X axis

acc_self_test_en_x		
0x00	disabled	disabled
0x01	enabled	enabled

acc_self_test_en_y: trigger self test for Y axis

acc_self_test_en_y		
0x00	disabled	disabled
0x01	enabled	enabled

acc_self_test_en_z: trigger self test for Z axis

acc_self_test_en_z		
0x00	disabled	disabled
0x01	enabled	enabled

acc_self_test_sign: select sign of self-test excitation

acc_self_test_sign		
0x00	positive	positive
0x01	negative	negative

Register (0x7E) CMD

DESCRIPTION: Command Register
RESET: 0x00
DEFINITION (Go to register map):

Name	**Register (0x7E) CMD**			
Bit	7	6	5	4
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	cmd			
Bit	3	2	1	0
Read/Write	RW	RW	RW	RW
Reset Value	0	0	0	0
Content	cmd			

cmd: Available commands (Note: Register will always read as 0x00):

cmd		
0x00	nop	reserved. No command.
0xb0	fifo_flush	Clears all data in the FIFO, does not change FIFO_CONFIG and FIFO_DOWNS registers
0xb1	step_cnt_clear	Clears the value of the step counter to 0
0xb6	softreset	Triggers a reset, all user configuration settings are overwritten with their default state

The device supports a command set to trigger certain activities and state transitions of the device. The command interpreter is connected to register *cmd*. A command is invoked if the corresponding Opcode is written to the *cmd* register.

Writing an undefined command to register *cmd* has no effect and ERR_REG.*cmd_err*=’0’ in this case.

The device implements a simple handshaking mechanism to signal its readiness for accepting a new command. Prior to writing a new command to the *cmd* register the user must read the status bit STATUS.*cmd_rdy*:

cmd_rdy = ’1’: device is ready to accept a command

cmd_rdy = ’0’: a command is being executed, any new command is ignored

cmd_err is set to ‘1’ when command execution failed. *cmd_err* is reset to ‘0’ if the last command execution was successful. This is a clear-on-read bit.

After the *softreset* command has been invoked, status and error register are updated after *Tst_up*.

6. Digital Interfaces

6.1. Interface

By default, the BMA400 operates in I2C mode. The BMA400 interface can also be configured to operate in a SPI 4-wire configuration. It can also be re-configured by software to work in 3-wire mode instead of 4-wire mode.
All three possible digital interfaces share partly the same pins. The mapping for the primary interface of the BMA400 is given in the following table:

Pin#	Name	I/O Type	Description	Connect to (Primary IF)		
				in SPI4W	in SPI3W	in I2C
1	SDO	Digital I/O	Serial data output in SPI Address select in I²C mode see chapter 7.2	SDO	DNC (float)	GND for default I2C addr.
2	SDX	Digital I/O	SDA serial data I/O in I²C SDI serial data input in SPI 4W SDA serial data I/O in SPI 3W	SDI	SDA	SDA
5	INT1	Digital I/O	Interrupt output 1 (default)	INT1	INT1	INT1
6	INT2	Digital I/O	Interrupt output 2 (default)	INT2	INT2	INT2
10	CSB	Digital in	Chip select for SPI mode	CSB	CSB	V_{DDIO}
12	SCX	Digital in	SCK for SPI serial clock SCL for I²C serial clock	SCK	SCK	SCL

* If INT1 and/or INT2 are not used, please do not connect them (DNC).

The following table shows the electrical specifications of the interface pins:

Parameter	Symbol	Condition	Min	Typ	Max	Units
Pull-up Resistance, CSB pin	R_{up}	Internal Pull-up Resistance to VDDIO		120		kΩ
I²C Bus Load Capacitance (max. drive capability)	C_{I2C_Load}				400	pF

6.2. Interface I2C/SPI Protocol Selection

The protocol is automatically selected based on the chip select CSB pin behavior after power-up. At reset / power-up, BMA400 is in I2C mode. If CSB is connected to VDDIO during power-up and not changed the sensor interface works in I2C mode. For using I2C, it is recommended to hard-wire the CSB line to VDDIO. Since power-on-reset is only executed when both VDD and VDDIO are established, there is no risk of incorrect protocol detection due to power-up sequence.
If CSB sees a rising edge after power-up, the BMA400 interface switches to SPI until a reset or the next power-up occurs. Therefore, a CSB rising edge is needed before starting the SPI communication. Hence, it is mandatory to perform a SPI single read of e.g. register CHIP_ID (the obtained value will be invalid) before the actual communication start, in order to use the SPI interface.

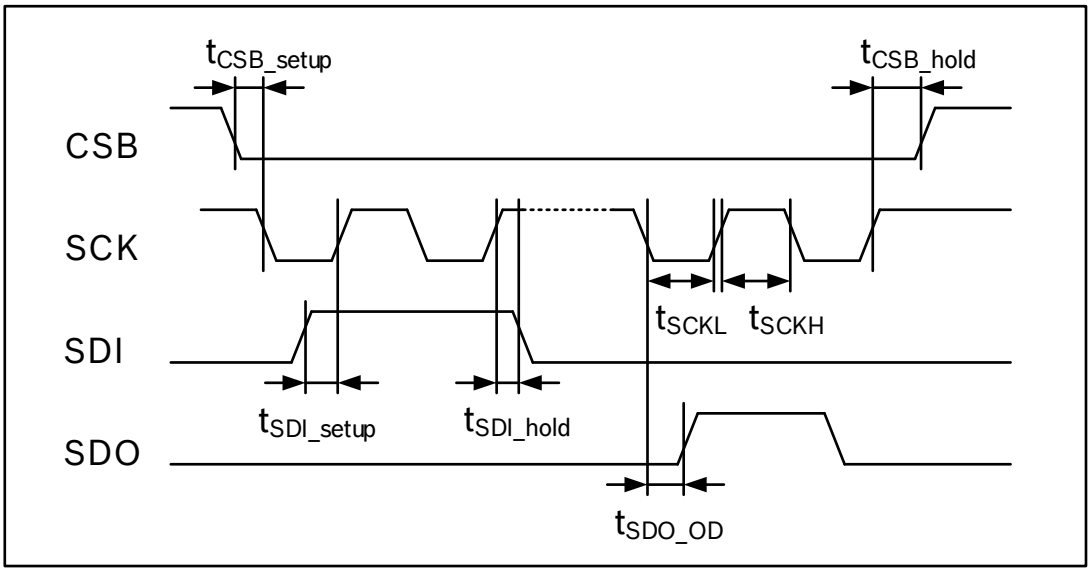
6.3. SPI interface and protocol

The timing specification for SPI of the BMA400 is given in the following table:

Following SPI timing table, valid at $V_{DDIO} \geq 1.71V$

Parameter	Symbol	Condition	Min	Max	Units
Clock Frequency	f_{SPI}	Max. Load on SDI or SDO = 25pF, $V_{DDIO} \geq 1.71$ V		17	MHz
		$V_{DDIO} < 1.71V$		7	MHz
SCK Low Pulse	t_{SCKL}		20		ns
SCK High Pulse	t_{SCKH}		20		ns
SDI Setup Time	t_{SDI_setup}		20		ns
SDI Hold Time	t_{SDI_hold}		20		ns
SDO Output Delay	t_{SDO_OD}	Load = 25pF, $V_{DDIO} \geq 1.71V$		30	ns
CSB Setup Time	t_{CSB_setup}		For first SPI transaction after reset (due to interface changed from I2C to SPI mode)	20 50	ns
CSB Hold Time	t_{CSB_hold}		20		ns

The following figure shows the definition of the SPI timings:

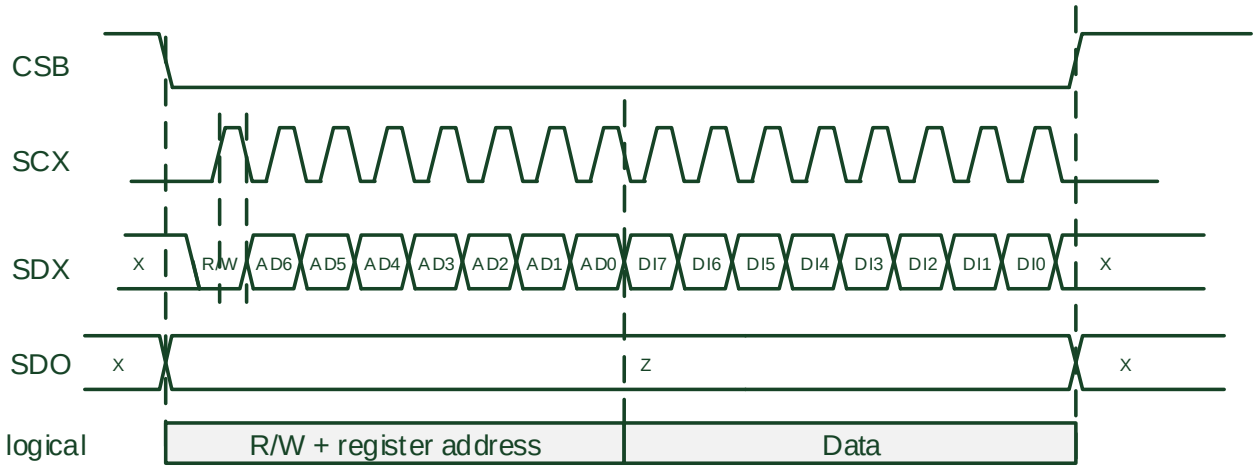


SPI timing diagram

The SPI interface of the BMA400 is compatible with two modes, ´00´ [CPOL = ´0´ and CPHA = ´0´] and ´11´ [CPOL = ´1´ and CPHA = ´1´]. The automatic selection between ´00´ and ´11´ is controlled based on the value of SCK after a falling edge of CSB.

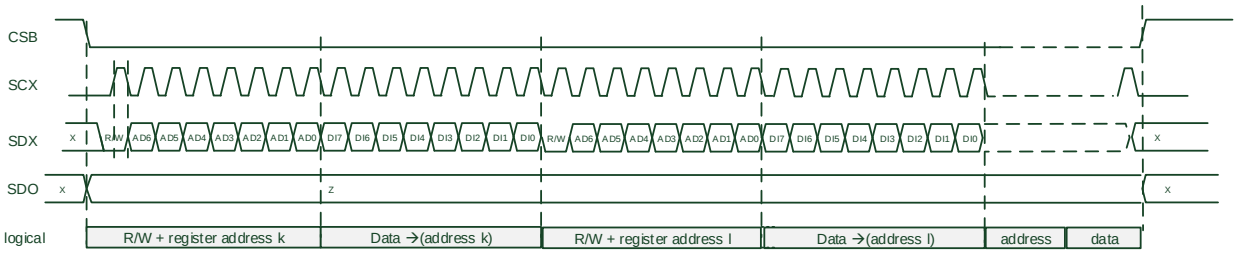
Two configurations of the SPI interface are supported by the BMA400: 4-wire and 3-wire. The same protocol is used by both configurations. The device operates in 4-wire configuration by default. It can be switched to 3-wire configuration by writing IF_CONF.spi3 = 0b1. Pin SDI is used as the common data pin in 3-wire configuration.
For single byte read as well as write operations, 8-bit protocols are used. The BMA400 also supports multiple-byte read and write operations.

In SPI 4-wire configuration CSB (chip select low active), SCK (serial clock), SDI (serial data input), and SDO (serial data output) pins are used. The communication starts when the CSB is pulled low by the SPI master and stops when CSB is pulled high. SCK is also controlled by SPI master. SDI and SDO are driven at the falling edge of SCK and should be captured at the rising edge of SCK.
The basic write operation waveform for 4-wire configuration is depicted in the following figure. During the entire write cycle SDO remains in high-impedance state.



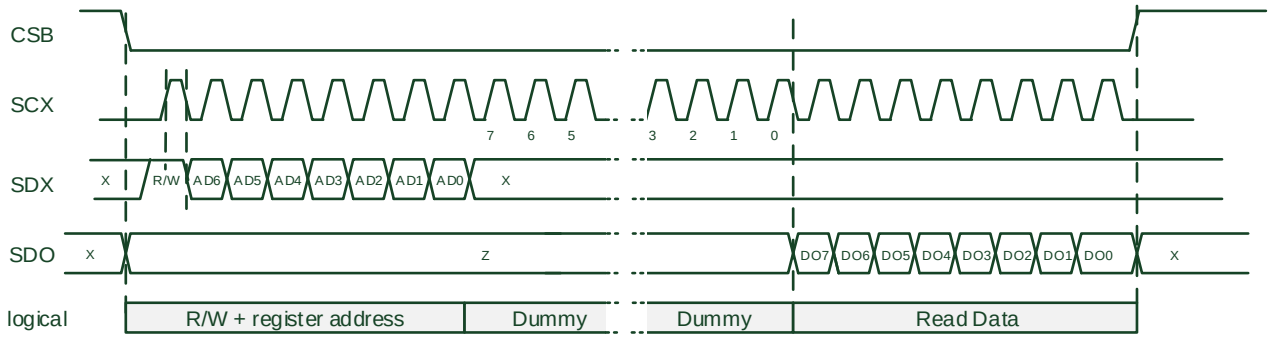
4-wire basic SPI write sequence (mode ´00´)

Multiple write operations are possible by keeping CSB low and continuing the data transfer. Every data must be preceded by R/W flag and address, there is no address auto-increment like in burst read mode. The principle of multiple write is shown in figure below:



SPI multiple write

The basic read operation waveform for 4-wire configuration is depicted in the figure below. Please note that the first byte received from the BMA400 via the SDO line correspond to a dummy byte and the 2nd byte correspond to the value read out of the specified register address. That means, for a basic read operation two bytes have to be read and the first has to be dropped and the second byte must be interpreted.



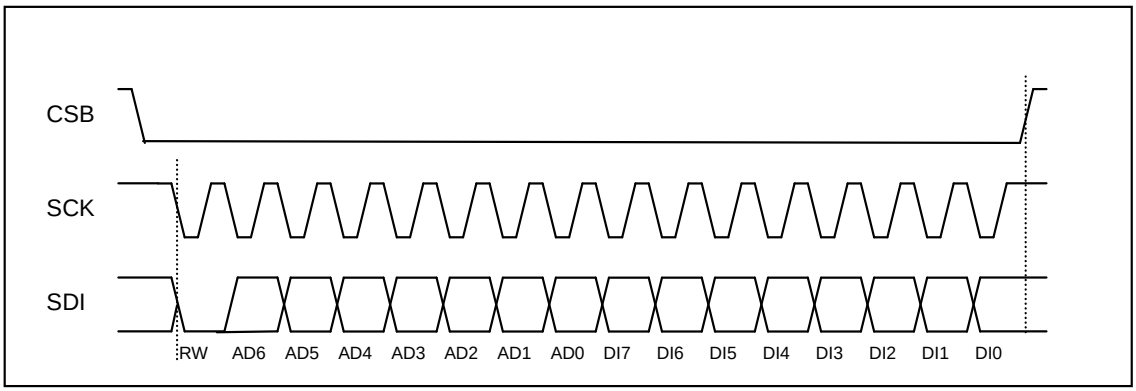
4-wire basic SPI read sequence (mode ´00´)

The data bits are used as follows:
R/W: Read/Write bit. When 0, the data SDI is written into the chip. When 1, the data SDO from the chip is read.
AD6-AD0: Register Address
DI7-DI0: When in write mode, these are the data SDI, which will be written into the address.
DO7-DO0: When in read mode, these are the data SDO, which are read from the address.
Multiple read operations are possible by keeping CSB low and continuing the data transfer. Only the first register address has to be written. Addresses are automatically incremented after each read access as long as CSB stays active low. Please note that the first byte received from the BMA400 via the SDO line correspond to a dummy byte and the 2nd byte correspond to the value read out of the specified register address. The successive bytes read out correspond to values of incremented register addresses. That means, for a multiple read operation of n bytes, n+1 bytes have to be read, the first has to be dropped and the successive bytes must be interpreted. When reaching address FIFO_DATA, auto-increment stops, and the FIFO is read bytewise.

In SPI 3-wire configuration CSB (chip select low active), SCK (serial clock), and SDA (serial data input and output) pins are used. While SCK is high, the communication starts when the CSB is pulled

low by the SPI master and stops when CSB is pulled high. SCK is also controlled by SPI master. SDA is driven (when used as input of the device) at the falling edge of SCK and should be captured (when used as the output of the device) at the rising edge of SCK.

The protocol as such is the same in 3-wire configuration as it is in 4-wire configuration. The basic operation wave-form (read or write access) for 3-wire configuration is depicted in the figure below:



3-wire basic SPI read or write sequence (mode ´11´)

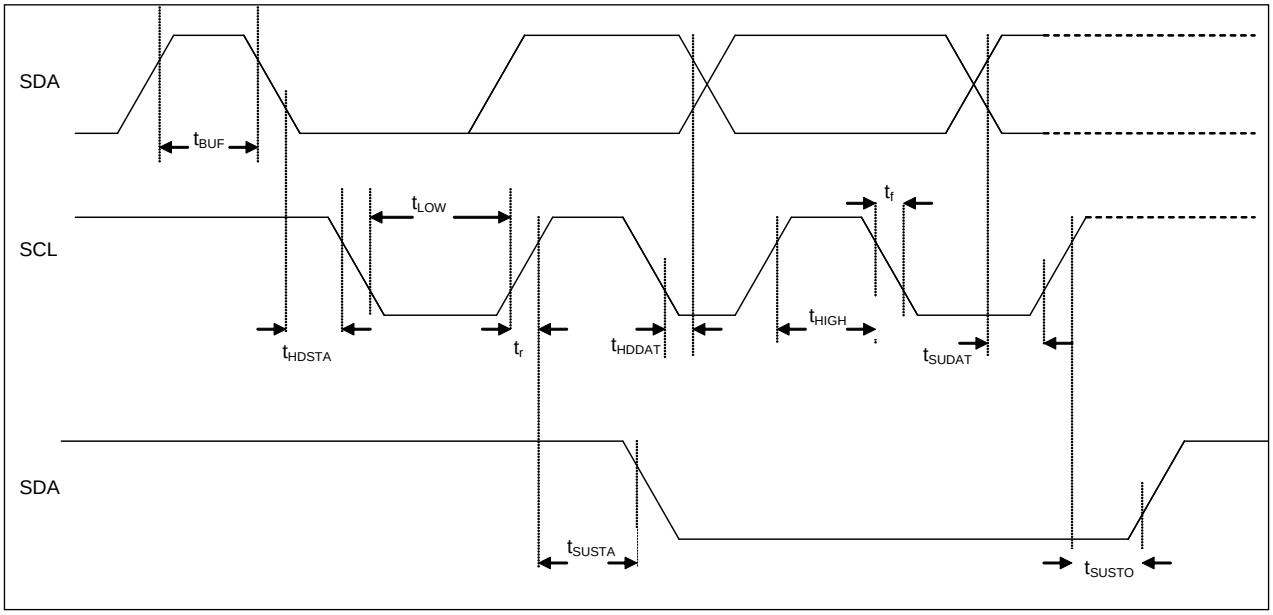
6.4. Primary I2C Interface

The I²C bus uses SCL (= SCx pin, serial clock) and SDA (= SDx pin, serial data input and output) signal lines. Both lines Must be connected to V_{DDIO} externally via pull-up resistors so that they are pulled high when the bus is free.
The default I²C address of the device is b001010X . 'X' is defined by the SDO pin: if SDO pulled to ´GND´ X equals 0, is SDO is pulled to VDDIO, X equals 1. In I2C, the SDO level must be defined, it cannot be left floating.
The I²C interface of the BMA400 is compatible with the I²C Specification UM10204 Rev. 06 (April 2014), available at http://www.nxp.com. The BMA400 supports **I²C standard mode and fast mode**, only 7-bit address mode is supported. For V_{DDIO} = 1.2V to 1.62 V the guaranteed voltage output levels are slightly relaxed as described in Table 1 of the electrical specification section.
BMA400 also supports an **extended I²C mode** that allows using clock frequencies up to 3.4 MHz. In this mode all timings of the fast mode apply and it additionally supports clock frequencies up to 3.4MHz.

The timing specification for I²C of the BMA400 is given in the following table:

Parameter	Symbol	Condition	Min	Max	Units
Clock Frequency	f_{SCL}			3400	kHz
SCL Low Period	t_{LOW}		1.3		µs
SCL High Period	t_{HIGH}		0.6		
SDA Setup Time	t_{SUDAT}		0.1		
SDA Hold Time	t_{HDDAT}		0.0		
Setup Time for a repeated Start Condition	t_{SUSTA}		0.6		
Hold Time for a Start Condition	t_{HDSTA}		0.6		
Setup Time for a Stop Condition	t_{SUSTO}		0.6		
Time before a new Transmission can start	t_{BUF}	normal mode	1.3		

The figure below shows the definition of the I²C timings



I²C timing diagram

The I²C protocol works as follows:

START: Data transmission on the bus begins with a high to low transition on the SDA line while SCL is held high (start condition (S) indicated by I²C bus master). Once the START signal is transferred by the master, the bus is considered busy.

STOP: Each data transfer should be terminated by a Stop signal (P) generated by master. The STOP condition is a low to high transition on SDA line while SCL is held high.

ACKS: Each byte of data transferred must be acknowledged. It is indicated by an acknowledge bit sent by the receiver. The transmitter must release the SDA line (no pull down) during the acknowledge pulse while the receiver must then pull the SDA line low so that it remains stable low during the high period of the acknowledge clock cycle.

In the following diagrams these abbreviations are used:

S	Start
P	Stop
ACKS	Acknowledge by slave
ACKM	Acknowledge by master
NACKM	Not acknowledge by master
RW	Read / Write

A START immediately followed by a STOP (without SCL toggling from ´VDDIO´ to ´GND´) is not supported. If such a combination occurs, the STOP is not recognized by the device.

I²C write access:

I²C write access can be used to write a data byte in one sequence.

The sequence begins with start condition generated by the master, followed by 7 bits slave address and a write bit (RW = 0). The slave sends an acknowledge bit (ACKS = 0) and releases the bus. Then the master sends the one byte register address. The slave again acknowledges the transmission and waits for the 8 bits of data which shall be written to the specified register address. After the slave acknowledges the data byte, the master generates a stop signal and terminates the writing protocol.

Example of an I²C write access:

Start	Slave Adress							R/W	ACK		Register address (0x41)							ACK	Register data (0x01)								ACK	Stop
	0	0	1	0	1	0	S	0		L	1	0	0	0	0	0	1		0	0	0	0	0	0	0	1		

Master -> Slave
Slave -> Master

S defined by SDO
L: tie to LOW, not part of address

Multi-byte writes are supported without restriction on normal registers.

Start	Slave Adress							R/W	ACK		Register address (0x41)							ACK	Register data (0x01)								ACK
	0	0	1	0	1	0	S	0		L	1	0	0	0	0	0	1		0	0	0	0	0	0	0	1	

Master->Slave
Slave->Master

	Register address (0x42)							ACK	Register data (0x01)								ACK	Stop
L	1	0	0	0	0	1	0		0	0	0	0	0	0	0	1		

I²C read access:

I²C read access also can be used to read one or multiple data bytes in one sequence.
A read sequence consists of a one-byte I²C write phase followed by the I²C read phase. The two parts of the transmission must be separated by a repeated start condition (S). The I²C write phase addresses the slave and sends the register address to be read. After slave acknowledges the transmission, the master generates again a start condition and sends the slave address together with a read bit (RW = 1). Then the master releases the bus and waits for the data bytes to be read out from slave. After each data byte the master has to generate an acknowledge bit (ACKS = 0) to enable further data transfer. A NACKM (ACKS = 1) from the master stops the data being transferred from the slave. The slave releases the bus so that the master can generate a STOP condition and terminate the transmission.
The register address is automatically incremented and, therefore, more than one byte can be sequentially read out. Once a new data read transmission starts, the start address will be set to the register address specified since the latest I²C write command. By default the start address is set at 0x00. In this way repetitive multi-bytes reads from the same starting address are possible.

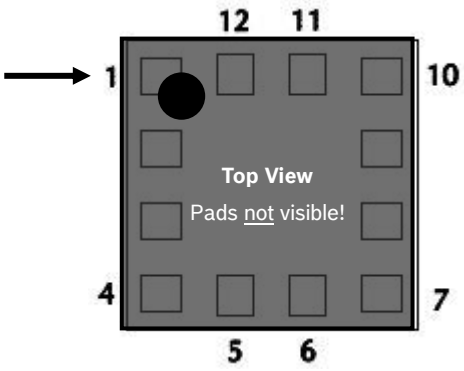
Start	Slave Adress							R/W	ACK		Register address (0x05)							ACK
	0	0	1	0	1	0	S	**0**		L	0	0	0	0	1	0	1	

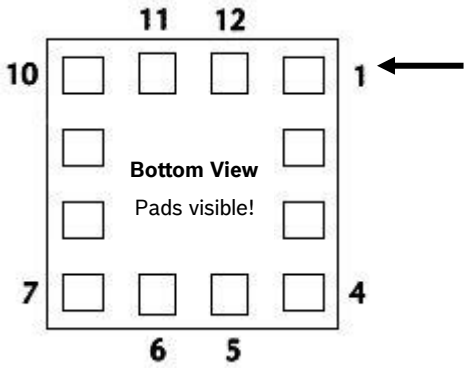
Start	Slave Adress							R/W	ACK	Register data - address 0x05								ACK	Register data - address 0x06								NACK	Stop
	0	0	1	0	1	0	S	**1**		d7	d6	d5	d4	d3	d2	d1	d0		d7	d6	d5	d4	d3	d2	d1	d0		

Master->Slave
Slave->Master

7. Pin-out and Connection Diagrams

7.1. Pin-out





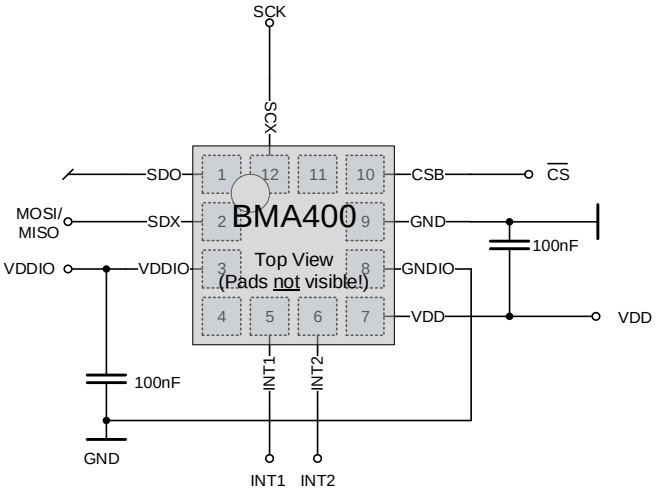
Pin description

Pin#	Name	I/O Type	Description	Connect to		
				in SPI 4W	In SPI 3W	in I²C
1	SDO	Digital I/O	Serial data output in SPI Address select in I²C mode see chapter 7.2	SDO	DNC (float)	GND for default I2C addr.
2	SDX	Digital I/O	SDA serial data I/O in I²C SDI serial data input in SPI 4W SDA serial data I/O in SPI 3W	SDI	SDA	SDA
3	VDDIO	Supply	Digital I/O supply voltage (1.2V … 3.6V)	V_{DDIO}	V_{DDIO}	V_{DDIO}
4	NC					
5	INT1	Digital I/O	Interrupt output 1 (default)	INT1	INT1	INT1
6	INT2	Digital I/O	Interrupt output 2 (default)	INT2	INT2	INT2
7	VDD	Supply	Power supply for analog & digital domain (1.62V … 3.6V)	V_{DD}	V_{DD}	V_{DD}
8	GNDIO	Ground	Ground for I/O	GND	GND	GND
9	GND	Ground	Ground for digital & analog	GND	GND	GND
10	CSB	Digital in	Chip select for SPI mode	CSB	CSB	V_{DDIO}
11	NC					
12	SCX	Digital in	SCK for SPI serial clock SCL for I²C serial clock	SCK	SCK	SCL

7.2. Connection Diagrams

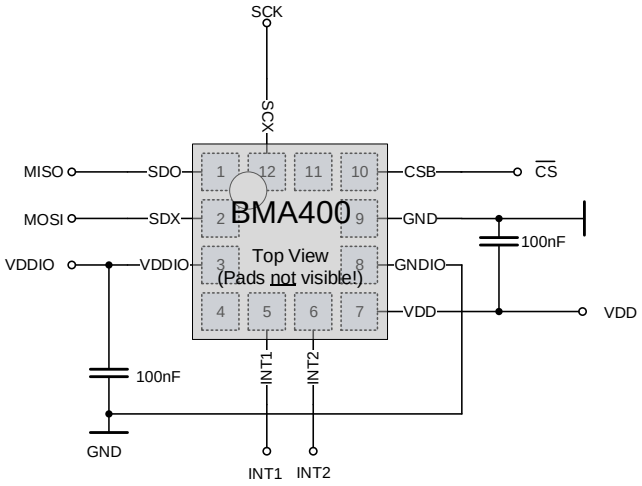
SPI

3-wire



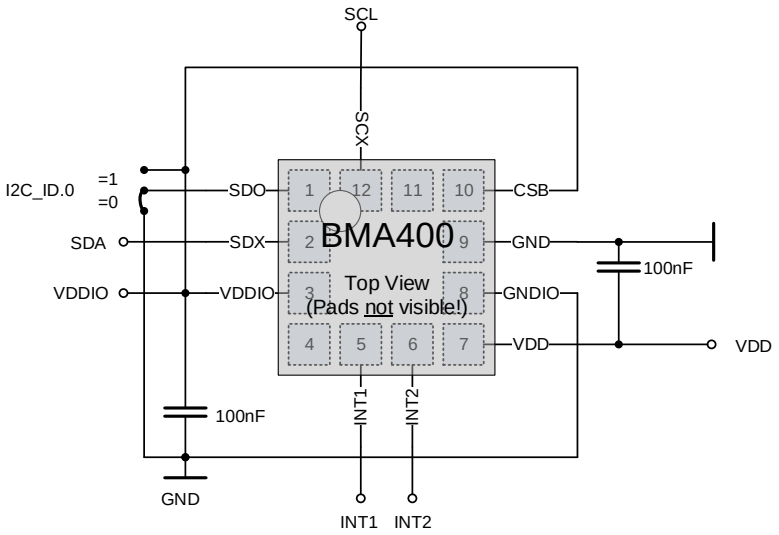
It is recommended to use 100nF decoupling capacitors at pin 3 (VDDIO) and pin 7 (VDD).

4-wire



It is recommended to use 100nF decoupling capacitors at pin 3 (VDDIO) and pin 7 (VDD).

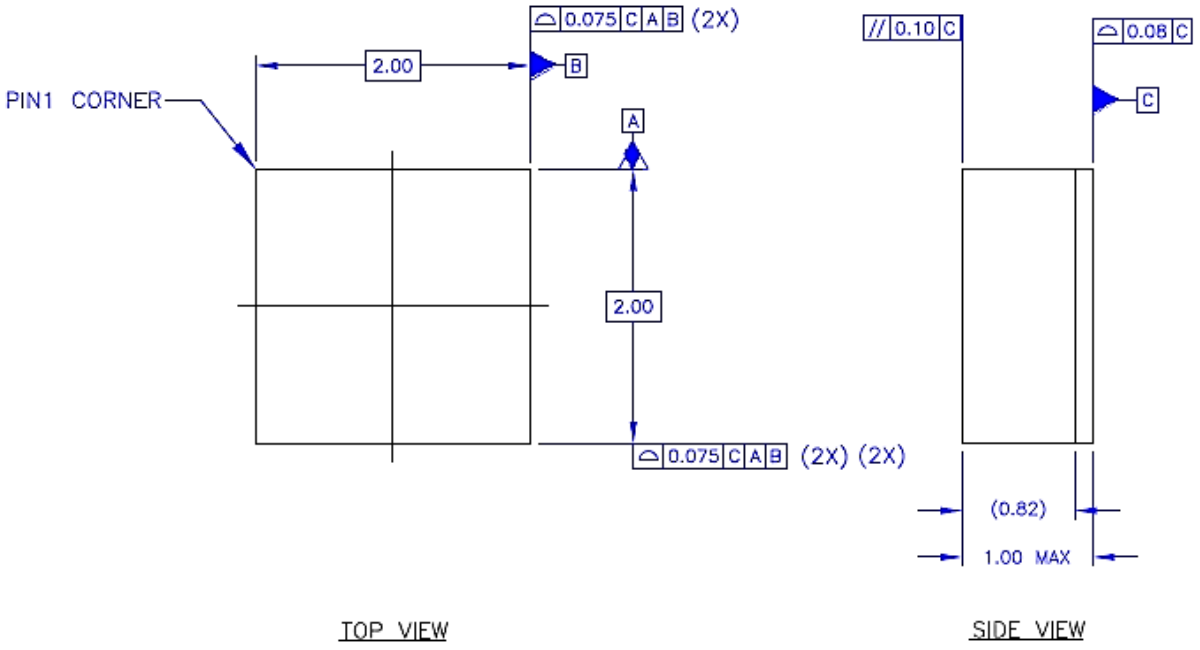
I2C

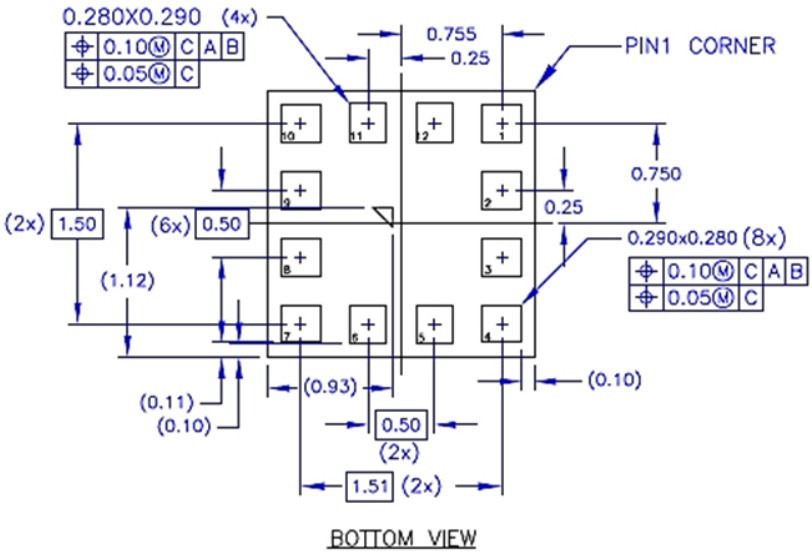


It is recommended to use 100nF decoupling capacitors at pin 3 (VDDIO) and pin 7 (VDD).

8. Package

8.1. Package outline dimensions



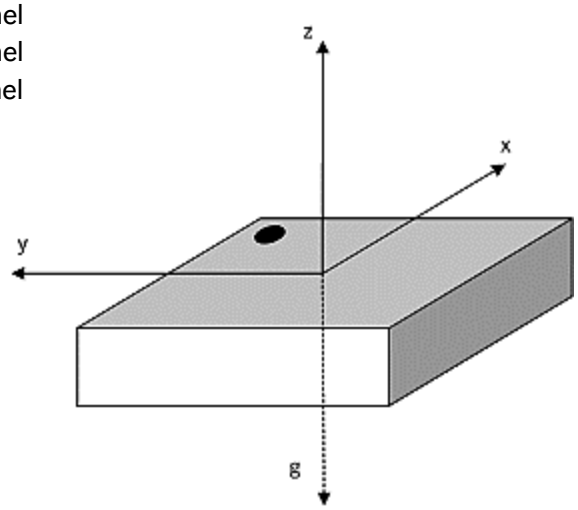


8.2. Sensing axis orientation

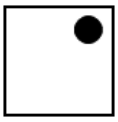
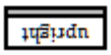
If the sensor is accelerated in the indicated directions, the corresponding channel will deliver a positive acceleration signal (dynamic acceleration). If the sensor is at rest and the force of gravity is acting along the indicated directions, the output of the corresponding channel will be negative (static acceleration).

Example: If the sensor is at rest or at uniform motion in a gravity field according to the figure given below, the output signals are:

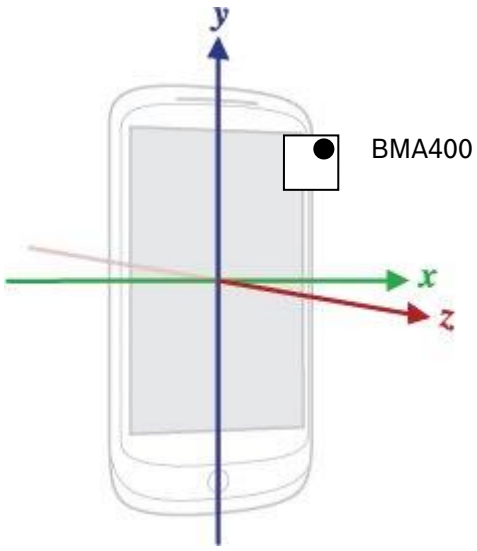
- ± 0g for the X channel
- ± 0g for the Y channel
- + 1g for the Z channel



The following table lists all corresponding output signals on X, Y, and Z while the sensor is at rest or at uniform motion in a gravity field under assumption of a ±4g range setting, a 16 bit resolution, and a top down gravity vector as shown above.

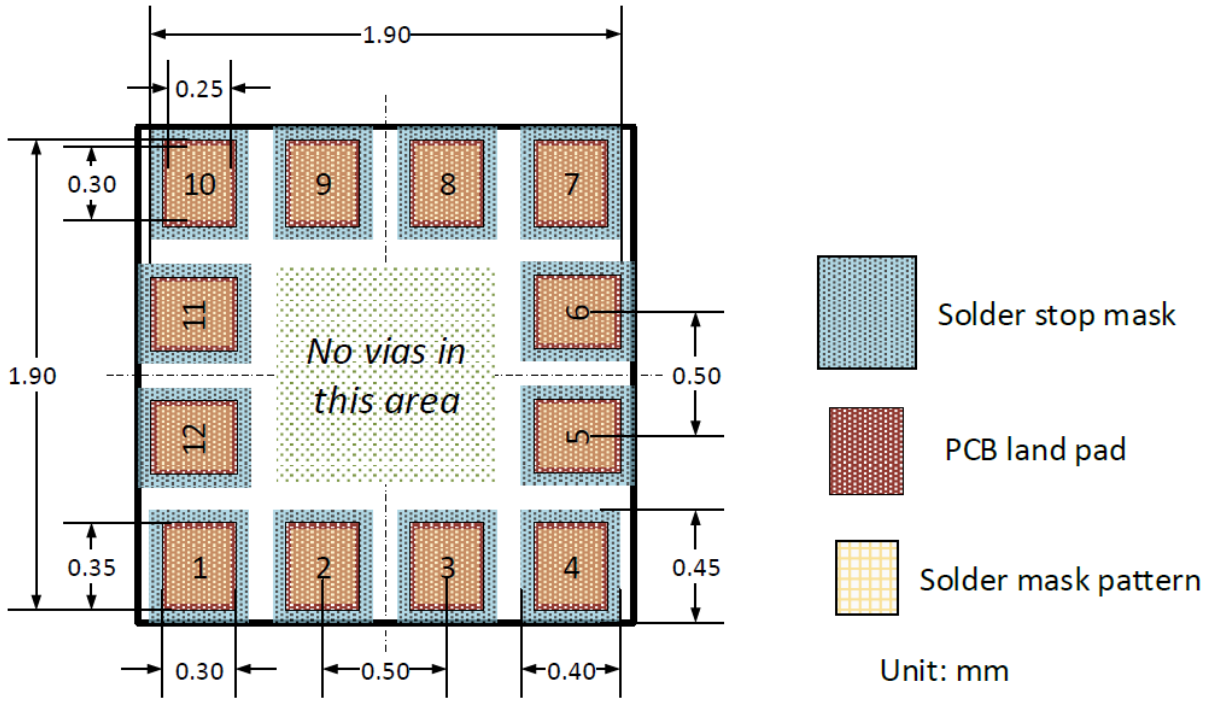
Sensor Orientation (gravity vector ↓)						
Output Signal X	0g / 0 LSB	1g / 1024 LSB	0g / 0 LSB	-1g / -1024 LSB	0g / 0 LSB	0g / 0 LSB
Output Signal Y	-1g / -1024 LSB	0g / 0 LSB	1g / 1024 LSB	0g / 0 LSB	0g / 0 LSB	0g / 0 LSB
Output Signal Z	0g / 0 LSB	0g / 0 LSB	0g / 0 LSB	0g / 0 LSB	1g / 1024 LSB	-1g / -1024 LSB

For reference the figure below shows the device orientation with an integrated BMA400.



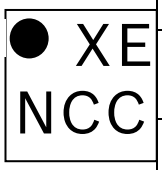
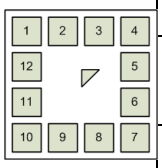
8.3. Landing pattern recommendation

The recommended landing pattern for the BMA400 on customer's PCB is given in the following figure. It is recommended to avoid any wiring underneath the BMA400 (shaded area).

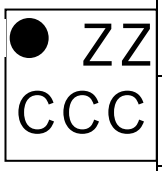
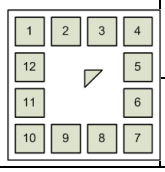


8.4. **Marking**

Marking of Engineering Samples(A,C) BMA400

Labeling	Name	Symbol	Remark
Top view 	Sub-con ID	X	internal use only
	Eng. sample ID	E	Identifies Engineering Samples
	Sample ID	NCC	'N' to be replaced by 'A','C', sample status 'CC' defines lot number
Bottom view 	Pin 1 identifier top side	●	--
	Pin 1 identifier bottom side	◤	Triangle points in the direction of pin 1

Marking of Mass Production Samples BMA400

Labeling	Name	Symbol	Remark
Top view 	Supply chain ID	ZZ	internal use only
	Counter ID	CCC	3 alphanumeric digits, variable to generate trace-code.
Bottom view 	Pin 1 identifier top side	●	--
	Pin 1 identifier bottom side	◤	Triangle points in the direction of pin 1

8.5. Soldering guidelines

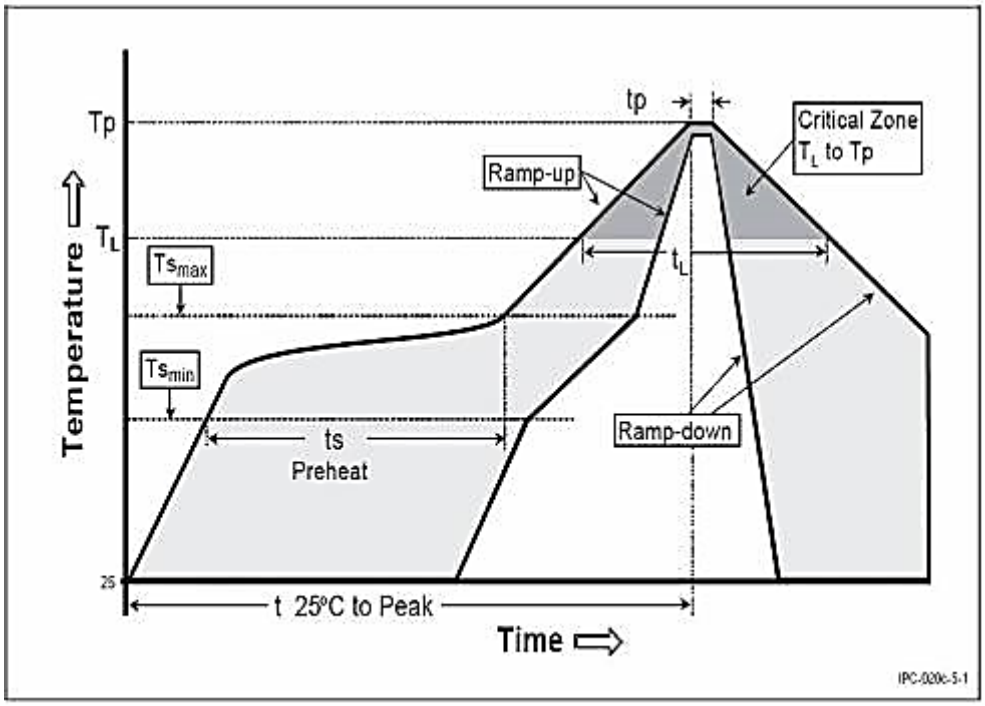
The moisture sensitivity level of the BMA400 sensors corresponds to JEDEC Level 1, see also

- *IPC/JEDEC J-STD-020E "Joint Industry Standard: Moisture/Reflow Sensitivity Classification for non-hermetic Solid State Surface Mount Devices"*
- *IPC/JEDEC J-STD-033D "Joint Industry Standard: Handling, Packing, Shipping and Use of Moisture/Reflow Sensitive Surface Mount Devices"*

The sensor fulfils the lead-free soldering requirements of the above-mentioned IPC/JEDEC standard, i.e. reflow soldering with a peak temperature up to 260°C.

Profile Feature		Pb-Free Assembly
Average Ramp-Up Rate (Ts_{max} to Tp)		3° C/second max.
Preheat – Temperature Min (Ts_{min}) – Temperature Max (Ts_{max}) – Time (ts_{min} to ts_{max})		150 °C 200 °C 60-180 seconds
Time maintained above: – Temperature (T_L) – Time (t_L)		217 °C 60-150 seconds
Peak/Classification Temperature (Tp)		260 °C
Time within 5 °C of actual Peak Temperature (tp)		20-40 seconds
Ramp-Down Rate		6 °C/second max.
Time 25 °C to Peak Temperature		8 minutes max.

Note 1: All temperatures refer to topside of the package, measured on the package body surface.



8.6. Handling instructions

Micromechanical sensors are designed to sense acceleration with high accuracy even at low amplitudes and contain highly sensitive structures inside the sensor element. The MEMS sensor can tolerate mechanical shocks up to several thousand g's. However, these limits might be exceeded in conditions with extreme shock loads such as e.g. hammer blow on or next to the sensor, dropping of the sensor onto hard surfaces etc.

We recommend to avoid g-forces beyond the specified limits during transport, handling and mounting of the sensors in a defined and qualified installation process.

This device has built-in protections against high electrostatic discharges or electric fields (e.g. 2kV HBM); however, anti-static precautions should be taken as for any other CMOS component. Unless otherwise specified, proper operation can only occur when all terminal voltages are kept within the supply voltage range. Unused inputs must always be tied to a defined logic voltage level.

8.7. Environmental safety

The BMA400 sensor meets the requirements of the EC restriction of hazardous substances (RoHS) directive, see also:

RoHS – Directive 2011/65/EU and its amendments, including the amendment 2015/863/EU on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Halogen content

The BMA400 is halogen-free. For more details on the corresponding analysis results please contact your Bosch Sensortec representative.

Internal package structure

Within the scope of Bosch Sensortec's ambition to improve its products and secure the mass product supply, Bosch Sensortec qualifies additional sources (e.g. 2nd source) for the LGA package of the BMA400.

While Bosch Sensortec took care that all of the technical packages parameters are described above are 100% identical for all sources, there can be differences in the chemical content and the internal structural between the different package sources.

However, as secured by the extensive product qualification process of Bosch Sensortec, this has no impact to the usage or to the quality of the BMA400 product.

9. Legal disclaimer

9.1. Engineering samples

Engineering Samples are marked with an asterisk (*), (E) 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.

9.2. Product use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product data sheet. They are not fit for use in life-sustaining or safety-critical systems. Safety-critical systems are those for which a malfunction is expected to lead to bodily harm, death or severe property damage. In addition, they shall not be used directly or indirectly for military purposes (including but not limited to nuclear, chemical or biological proliferation of weapons or development of missile technology), nuclear power, deep sea or space applications (including but not limited to satellite technology).
Bosch Sensortec products are released on the basis of the legal and normative requirements relevant to the Bosch Sensortec product for use in the following geographical target market: BE, BG, DK, DE, EE, FI, FR, GR, IE, IT, HR, LV, LT, LU, MT, NL, AT, PL, PT, RO, SE, SK, SI, ES, CZ, HU, CY, US, CN, JP, KR, TW. If you need further information or have further requirements, please contact your local sales contact.
The resale and/or use of Bosch Sensortec products are at the purchaser's own risk and his own responsibility. The examination of fitness for the intended use is the sole responsibility of the purchaser. 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 to inform Bosch Sensortec without delay of all safety-critical incidents.

9.3. 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.

10. Document history and modification

Rev. No	Chapter	Description of modification/changes	Date
1.1	-	Public release	31 July 2018
1.2	4.8, 4.7, 8.3, 9.2	Update self-test limits Step Counter Configuration Landing Pattern Disclaimer updated	July 2019
1.3	2 8.5, 8.7	Updated absolute max rating for V_{DD} and V_{DDIO} Updated references	February 2020
1.4	4.7, 5.1 4.8 5.1	Removed stability mode of orientation Corrected typos Updated content: Register (0x7D) SELF_TEST	July 2020
1.5	9	Disclaimer update	November 2020
1.6	4.4, 5.1	Updated the BW filt1 and filt2 information	April 2021

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

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

Modifications reserved
Specifications subject to change without notice
Document number: BST-BMA400-DS000-07
Revision_1.6_042021